VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vzájemné transformace notací pro business modelování

Transformations of Business Process Notations

Akademický rok: 2009/2010 Bc. Lukáš Ščurek

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. 5. 2010

……..

Bc. Lukáš Ščurek

Chtěl bych poděkovat Ing. Statopluku Štolfovi, Ph.D. za připomínky a návrhy po celou dobu realizace diplomové práce.

Abstrakt:

Tato práce se zabývá problematikou transformací mezi notacemi, které jsou určeny k business modelování. Úkolem této práce je nastudovat oblast business modelování a notací, které se k modelování využívají.

Praktická část je věnována vývoji aplikace, jejichž funkcí bude převoditelnost mezi jednotlivými notacemi, a to prostřednictvím nadefinovaných převodních pravidel za pomocí metamodelů k jednotlivým notacím. Meta-modelové diagramy, se kterými bude aplikace pracovat, budou ve formátu XML schéma a ukázkové modely budou uložené pomocí XML.

Klíčová slova: Business modelování, diagram, notace, transformace, XML Schema

Abstract:

This thesis deals with transformations between notations, which are intended for business modeling. The goal of this work is to study the business model and notations which are used for modeling of business process.

The practical part is devoted to developing the application, the function will be transferable between different notations. Will be define rules of transfer through using meta-models for each notation. Meta-model diagrams which the application will work will be in the XML schema and samples of model are stored in XML.

Key words: Business modeling, diagram, notation, transformation, XML Schema

Seznam použitých symbol a zkratek

BPM	Business Process Modeling
BPR	Business Process Reengineering
API	Application Programming Interface
BPMN	Business Process Modeling Notation
BPMI	Business Process Management Initiative
BPEL	Business Process Execution Language
BPML	Business Process Modeling Language
BPD	Business Process Diagram
UML	Unified Modeling Language
IDEF	Integration DEFinition
eEPC	extended Event – Driven Process Chain
MOF	Meta Object Facility
OMG	Object Management Group
XML	Extensible Markup Language
XMI	XML Metadata Interchange
COMMA	Common Meta Model Architecture
GOPRR	Graph-Object-Property-Role-Relationship
IS	Information Systém (Informační Systém)
ERP	Enterprise Ressource Planning
WFM	WorkFlow Management
ISO	International Organization for Standardization

Obsah

Seznam obrázků a tabulek

1 Úvod

Business modelování je velice rozsáhlá a zajímavá oblast, která se stále neuvěřitelně rychle vyvíjí, hlavně ve směru, aby byla co nejjednodušší, nejintuitivnější, schopná zaznamenat dění reálného světa a zároveň byla jasně definovaná a použitelná ve světě informačních technologií. Tato metoda slouží k definování a zaznamenání všech procesů a aktivit, které se ve firmě či společnosti odehrávají a následně slouží k monitorování, analýze či vylepšení jednotlivých procesů firmy a to z důvodu, že každá firma chce být konkurence schopná, odvádět kvalitní práci bez zbytečných finančních ztrát, ale hlavně se rozvíjet a profitovat. Ovšem business modelování můžeme také propojit s IT technologiemi a využít k realizaci informačního systému, kdy tato metoda slouží mimo jiné jako nástroj komunikace mezi obchodníky, business analytiky, IT developery a dalšími osobami, které jsou zainteresovány do vývoje aplikace.

Táto práce se v první části bude zabývat teoretickým rozborem business procesů a studií čtyřech typů velmi používaných notací, které se k business modelování používají. Zcela konkrétně půjde o UML, EPC, BPMN a Petriho sítě. Dále se v teoretické části zaměříme na oblast metamodelování, která s metodou business modelování úzce souvisí a navíc budeme metamodelování využívat v praktické části této práce.

Praktická část této práce se bude zaměřovat na realizaci transformací mezi jednotlivými výše zmiňovanými notacemi. Z každé notace využijeme nejpoužívanější elementy, ze kterých sestrojíme odpovídající metamodel. Pro realizaci metamodelů využijeme třídní diagram z notace UML. Pro definování jednotlivých převodních pravidel sestavíme transformační mezi-metamodel, kdy každá notace bude převoditelná na tento mezi-metamodel a opačně. Namodelované metamodely převedeme do formátu XML Schema. S využitím programovacího jazyka Java vytvoříme aplikaci, které bude demonstrovat jednotlivé transformace.

2 Teoretická část

2.1 Modelování business procesů (BPM)

Každá firma, podnik či společnost vykonává aktivity, které jsme schopni identifikovat a popsat, ovšem s pomocí nástrojů informační technologie, jsme schopni tyto aktivity namodelovat. Těmto aktivitám říkáme business procesy. Množina těchto business procesů nám popisuje chování celé firmy. V systémovém a softwarovém inženýrství je korektní specifikace, analýza a následné namodelování základním předpokladem pro např. efektivní implementaci a nasazení vlastního informačního systému.

Business modelování, které je většinou realizováno obchodními analytiky a manažery, má také důležitou úlohu k napomáhání k efektivním změnám jednotlivých procesů, které by měly vést ke zlepšení chodu firmy, může mít za následek zjednodušení některého procesu, ušetření času či financí a tak přispět k celkovému zkvalitnění a růstu firmy v budoucnosti.

Modelování business procesů hraje důležitou roli při řízení podnikových procesů (BPM). Ale je třeba si uvědomit, že i když modelování business procesů a řízení podnikových procesů sdílí stejnou zkratku (BPM), jedná se o rozdílnou činnost, která bývá často mezi sebou zaměňována.

BPM se zabývá aspekty architektury podnikových procesů, což vede ke všem architekturám, které podnik zahrnuje. Vztahy podnikových procesů v kontextu s ostatními podnikovými systémy (např. datové architektury, organizační struktura, strategie, atd.), umožňují větší možnosti při analýze a plánování podnikových změn. Například při sloučení dvou firem je důležité pochopit procesy obou společností do detailu tak, že je management může správně a efektivně identifikovat a vyhnout se tak například propouštění z provozu.

Co se týče přístupu, tak business modelování nabízí tři základní abstrakce, které jsou využívány ve všech moderních metodách.

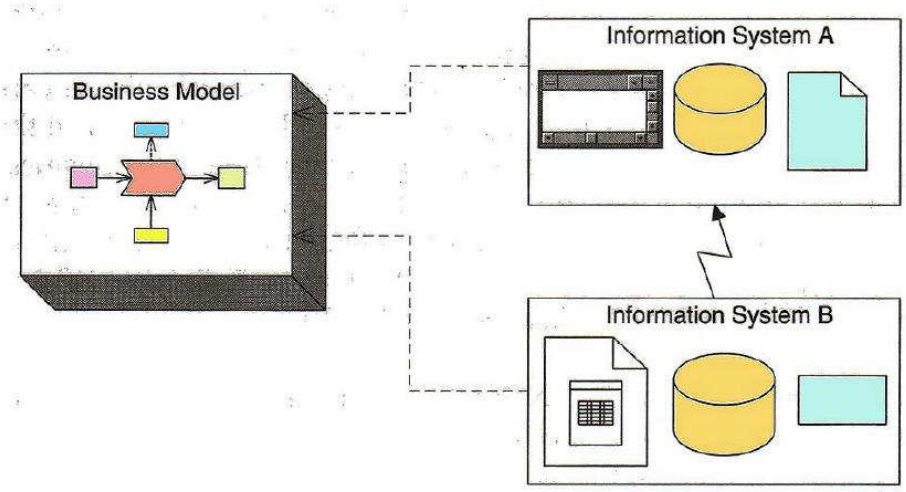
- ***funkční přístup*** *– je zaměřen především na funkce, jejich strukturování, vstupy a výstupy*
- ***přístup pomocí specifikací chování*** *– je specializován na řídící aspekty vykonávání procesu cestou stanovení událostí a podmínek, za kterých mohou být jednotlivé aktivity prováděny*
- ***strukturální přístup*** *– je určen k popisu statické struktury procesu. Jeho cílem je postihnout entity a zdroje vystupující v procesu, včetně jejich atributů, činností a vzájemných vazeb*

(*Převzato z www.uhk.panrepa.org/bpe_final/Kovarik_prace.doc****)***

2.1.1 Možnosti využití Business modelování

Business modelování je metoda, jak identifikovat aktivity, toky, procesy, které směřují k dosažení podnikatelského či strategického cíle. Přitom není zcela nutné vědět, jaké využití budou business modely ve finále mít. V praxi se ovšem tyto business modely realizují z níže uvedených důvodů.

- **Realizaci a zavedení IS** (Information System) - v tomto případě se tvorba business modelů stává počáteční fázi ve vývoji informačního systému. Tato metoda má hlavní výhodu při komunikaci se zadavatelem, na kterého je kladen požadavek, aby co nejpřesněji specifikoval i identifikoval skutečné procesy a úkony, které jsou ve firmě vykonávány a které se dále budou pomocí vytvářeného informačního systému realizovat. Z IT pohledu tvůrců takového softwarového díla má však business modelování daleko hlubší význam. Například navázání vytvořených modelů na další fáze softwarového procesu či automatizaci přechodů mezi jednotlivými fázemi.



Obrázek 1 - Zavedení IS

- **ERP systémy** (Enterprise Ressource Planning) – Pod tyto systémy spadají aplikační balíky jako např.: SAP či Oracle, pomocí kterých je možné automatizovat výrobní či podnikové procesy, finanční toky a řídit lidské zdroje. Pro realizaci těchto funkcí je třeba explicitně definovat patřičné procesy. Realizace takového ERP systému má podobný charakter jako u tvorby IS, kdy business modelování je základním stavebním kamenem na počátku softwarového procesu, kdy ovšem

výsledný produkt není IS, ale pouze funkčně a implementačně upravený či rozšířený stávající ERP systém.

- ***WFM systémy (***Workflow Management*).* Jedná se o systémy reprezentující obecné softwarové nástroje pro definici, realizaci a vlastní řízení podnikových procesů. V porovnání s ERP systémem se zde zaměřujeme přímo na podporu business procesu a WFM tak nabízejí obecnější možnosti pro práci s business procesy.
- **BPR** (Business Process Re-engineering) – Jemně odlišnou skupinu tvoří nástroje, které jsou označovány jako BPR. Ty slouží k vlastnímu modelování a analýze byznys procesů, kdy jejich cílem je umožnit a usnadnit vylepšování existujících či vytváření nových procesů. Takovéto procesy pak slouží k efektnějšímu řízení vlastního podniku či organizace. Této skupině je v této práci věnována samostatná kapitola.

Využití metody business modelování k jakémukoliv výše uvedenému účelu za pomocí jakéhokoliv nástroje, přináší řadu výhod, a to jak na úrovni vytváření software, tak na úrovni, kdy nám jde o dosažení podnikatelských cílů. Konkrétně jde o:

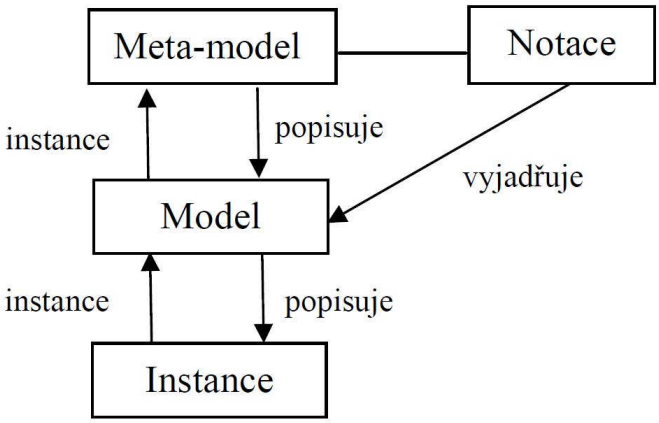
*- **snižování rizika při vývoji** – nabízí jednotný jazyk pro komunikaci mezi business analýzou a vlastním vývojem, díky němuž je možný snadný a jednoznačný přechod od požadavků k návrhu software*
*- **centralizovaný vývoj** – komplexní popis byznys procesů přináší možnost, kdy výsledná aplikace je vytvářena centralizovaně a není tvořena skládáním částí jednotlivých různých systémů*
*- **rychlý vývoj** – software je odproštěn od úkolu udržení cest účastníků procesu vedoucí k rychlejšímu vývoji a ke kódu, který je lépe udržovatelný*
*- **zvýšení efektivity** – automatizace business procesů vede k eliminaci mnoha zbytečných kroků*
*- **lepší řízení procesu** – vylepšený management business procesů dosažený standardizovanými pracovními metodami a dostupností auditačních cest*
*- **vylepšené zákaznické služby** – důslednost při vedení procesů vede k větší předvídatelnosti na úrovni odezvy k zákazníkům*
*- **pružnost (flexibilita)** – softwarové řízení procesů umožňuje jejich re-design ve smyslu změn podnikatelských požadavků*
*- **vylepšení business procesu** – soustředění na business procesy vede k jejich zefektivnění a zjednodušení*

(*Převzato z www.uhk.panrepa.org/bpe_final/Kovarik_prace.doc*)

2.1.2 Grafická reprezentace – Horizontální a Vertikální abstrakce

Grafické zobrazení informací obchodního procesu se ukázalo jako účinný prostředek pro komunikaci mezi zadavateli či obchodně-systémovými analytiky a vývojáři. Pro grafickou vizualizaci pomocí modelování využíváme modelovací notace resp. modelovací jazyky. Business modelování by se tedy mohlo obecně shrnout jako množina metod, přístupů a nástrojů používaných ke specifikaci a analýze business procesů a u využití některých nástrojů i případné následné simulaci.

Při modelování business procesů, neboli obecněji systémů, je využíván základní přístup, kdy je třeba identifikovat a následně oddělit jednotlivé úrovně, které budou modelovány. Dle využitého přístupu k modelování rozlišujeme rozdělení jednotlivých úrovní na horizontální a vertikální abstrakci. Relace mezi jednotlivými úrovněmi vertikálního modelu zachycuje Obr. 2.

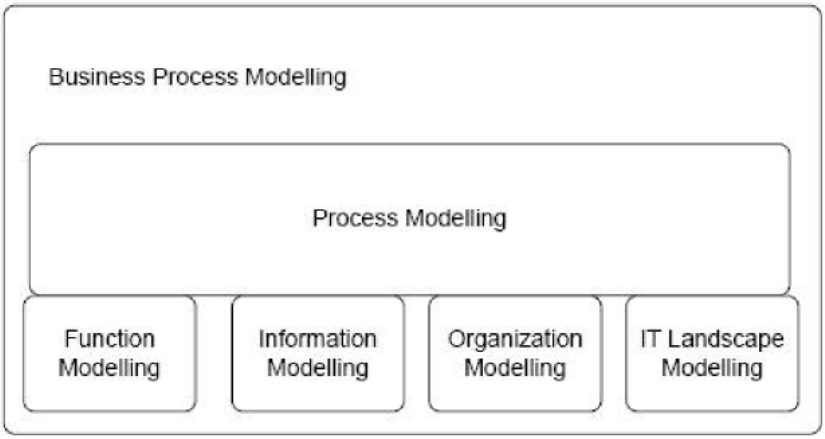


Obrázek 2 - Vertikální abstrakce

- ***Instance*** *- jedná se o nejnižší modelovací úroveň. Procesní instance představují konkrétní výskyt procesu. Do této úrovně spadají rovněž instance aktuálně prováděných aktivit vykonávaných v rámci procesu*
- ***Model*** *- identifikuje skupiny instancí procesu, které mají podobnou strukturu a usiluje o zachycení společného scénáře, podle kterého se řídí vykonávání aktivit těchto procesů. Jinými slovy řečeno se jedná o šablonu, která uvádí, jaké aktivity budou prováděny a v jakém pořadí se budou realizovat*

- ***Metamodel*** *– jedná se o úroveň s nejvyšším stupněm abstrakce, kdy meta-model slouží k zachycení konceptů a vztahů mezi nimi, které jsou použity při popisu modelu. S meta-modelem může být asociována určitá notace. Ta má obvykle grafickou podobu a slouží pro názornější a snadnější vyjádření vytvářeného modelu.*

(*Převzato z www.fit.vutbr.cz/study/courses/TJD/public/0910TJD-Macel.pdf*)



Obrázek 3 - Horizontální abstrakce

Vertikální abstrakce je nejznámější identifikace, která se zaměřuje na modelování dalších sub-domén podniku. Na obrázku jsou zachyceny sub-domény, které jsou důležité pro popis celkového obrazu podnikového procesu. Všechny uvedené modely sub-domén propojuje dohromady samotné procesní modelování.

- ***Funkční struktura*** *- modeluje organizaci práce uvnitř podniku, a proto je významná také pro ustanovení jednotlivých procesů*
- ***Informační struktura*** *- během procesu dochází k řadě rozhodnutí založených na datech. Jednotlivé aktivity procesu jsou datově provázány a model procesu musí zajistit, aby data potřebná pro jejich vykonávání byla dostupná*
- ***Organizační struktura*** *- umožňuje asociovat jednotlivé aktivity s určitými rolemi a tyto role zařadit do příslušných oddělení podniku*
- ***Struktura informačních technologií*** *- je často nezbytná pro podporu vykonávání aktivit procesu*

(Převzato z www.fit.vutbr.cz/study/courses/TJD/public/0910TJD-Macel.pdf)

2.1.3 Business Process Reengineering (BPR)

Jedná se o postup, který optimalizuje podnikové procesy ve vztahu k okolí tak, aby přinášely maximální efekty při optimální spotřebě podnikových zdrojů, a tím tak zvýšily konkurenceschopnost podniku. Výsledkem BPR je řada podstatných změn nejen v podnikových procesech, ale i v organizační a kvalifikační struktuře podniku, ve způsobech organizace práce a řízení.

Při reengineeringu se analyzuje, co se má udělat, kdo to má udělat, s čím, s jakými daty to má udělat, v jakých výstupních datech bude výsledek zachycen, na základě jaké události se co má vykonat, případně jaký aplikační software se při tom používá, jaká činnost navazuje atd. Hledáme pak optimální průběh procesu vzhledem ke spotřebovanému času a nákladům.

Třemi cílovými aspekty reengineeringu jsou:

- zaměření na procesy – pozornost by se měla zaměřit především na základní podnikatelské procesy, které se přímo týkají zákazníků, ne na procesy čistě interní. Základní procesy jsou pro úspěch v sektoru v němž organizace podniká rozhodující a firma by je měla identifikovat jako kritické, které rozhodnou o tom, zda přežije a porazí konkurenci
- radikální změna – reengineering neusiluje o změnu pro změnu. Cílem je konkurenceschopnost, a je-li to možné, ovládnutí trhu. Radikální změna je charakteristikou tohoto cíle a výsledkem akceptace procesního pohledu a opuštění staré cesty podnikání prostřednictvím funkčních oddělení
- dramatické zlepšení – zatímco od malých postupných zlepšení se dá eventuelně očekávat, že mohou mít velký kumulativní efekt, od reengineeringu se očekává dramatické zlepšení

2.1.4 Historie

Techniky pro modelování obchodních procesů, jako vývojový diagram, funkční blokové schéma toku, kontrolu diagram, Ganttův diagram, PERT diagram a IDEF se objevily na počátku 20. století.

- 1900 Ganttův diagram byl mezi prvními
- 1920 funkční Flow Blokové schéma
- 1950 PERT
- 1970 Data Flow Diagramy a IDEF

Pojem "modelování business procesů" byl vytvořen v roce 1960 v oblasti systémového inženýrství S. Williamsem. Jeho myšlenkou bylo, že metoda, kdy technik získává lepší pochopení fyzikálních kontrolních systémů, by mohla být použita podobným způsobem pro obchodní procesy. To trvalo až do roku 1990, než se termín stal populární.

V oblasti softwarového inženýrství termín "modelování podnikových procesů" stál proti software proces modelování, jehož cílem je zaměřit se více na stav praxe v průběhu vývoje softwaru. V této době všechny stávající i nové metody modelování podnikových procesů byly považovány za jazyky pro modelování business procesů. V objektově orientovaného přístupu, to bylo považováno za zásadní krok ve specifikaci Business Application Systems. Modelování business procesů se stalo základem nové metodiky, jak například také podpořit sběr dat, analýzu toků dat, diagramy procesů proudění a výkaznictví. Kolem roku 1995 byly prezentovány první vizuálně orientované nástroje pro modelování podnikových procesů a implementace.

2.1.5 Business model

Business model je prostředek pro vytváření hospodářské, sociální nebo jiné formy hodnot. Termín business model je používán pro širokou škálu formálních i neformálních reprezentací, které představují základní rysy podnikání, včetně účelu, nabídky, strategie, infrastruktury, organizační struktury, obchodních praktik či provozních procesů a zásad.

2.1.6 Business Process

Podnikové procesy je kolekce příbuzných, strukturovaných činností nebo úkolů, které produkují konkrétní službu nebo výrobek pro určitého zákazníka. Existují tři hlavní typy obchodních procesů:

1. Řídící procesy, které řídí provoz systému, například strategické řízení
2. Provozních procesy, které tvoří hlavní činnosti a vytváří primární hodnotu řetězce. Typické provozní procesy nákupu, výroby, marketingu a prodeje.
3. Podpůrné procesy, které podporují klíčové procesy. Mezi příklady patří účetnictví, technická podpora.

Podnikové procesy mohou být rozloženy do několika sub-procesů, které mají své vlastní atributy, ale také přispívají k dosažení cíle nadřazeného procesu. Analýza business procesů obvykle zahrnuje mapování procesů a sub-procesů až do úrovně elementárních aktivit a procedur.

Model business procesu je model z jednoho nebo více obchodních procesů, a definuje způsoby, jimiž se provádějí operace k dosažení uvedených cílů organizace. Obvykle v kontextu organizační struktury definující funkce rolí a jejich vztahy. Smyslem business modelování je tedy vytvořit abstrakci procesu, která umožňuje pochopení všech jeho aktivit, souvislostí mezi těmito aktivitami a rolemi reprezentovaných schopnostmi lidí a zařízení zapojených do daného procesu. Hlavní účel modelování podnikových procesů tedy spočívá ve vytvoření korektní specifikace business procesů a analýzy jejich vlastností.

Workflow je zobrazení posloupnosti operací, deklarované jako dílo člověka, dílo jednoduché, komplexní mechanismus, práce skupiny osob, práce organizace zaměstnanců nebo stroje. Workflow může být chápáno jako jakákoli abstrakce skutečné práce.

• **Aktivita** – Aktivita je popis činností a reprezentuje jeden atomický krok ve vykonávání procesu

• **Role** – Role je soubor vzájemně se doplňujících dovedností

• **Workflow** – Workflow je automatizovaný (pomocí výpočetní techniky) business proces

• **Model business procesu** – Model business procesu je abstraktní reprezentace business procesu, obvykle umožňující jeho další zpracování automatizovaným způsobem.

2.1.7 Nástroje pro modelování business procesů

Nástroje modelování business procesů poskytují obchodním uživatelům možnost modelovat své obchodní procesy, zavést a provádět tyto modely a zpřesnit tyto modely dle zpracovávaných dat. V důsledku toho mohou nástroje modelování business procesů zajistit transparentnost do podnikových procesů, centralizaci firemních modelů podnikových procesů a provádění měření.

Obchodní diagramy modelování procesů jsou:

- Use case diagramy vytvořené Ivar Jacobson, 1992. V současné době integrovány do UML
- Diagramy aktivit, v současné době rovněž přijala UML

Techniky modelování business procesů se kterými se budeme v této práci zabývat jsou:

- Business Process Modeling Notation (BPMN)
- Event-driven process chain (EPC)
- Petri Nets
- Unified Modeling Language (UML) – Aktivitní diagramy

Další příklady

- Cognition enhanced Natural language Information Analysis Method (CogNIAM)
- Extended Business Modeling Language (xBML)

2.1.8 BPM v praxi

Ve využití v praxi je daleko důležitější korektnost a úplnost popisovaného procesu před výběrem konkrétní použité modelovací metody. Důraz je kladen hlavně na co nepřesnější zachycení reálného světa a jeho přenesení do modelu daného procesu, což nemusí být v mnoha případech nejjednodušší. Dalším nelehkým úkonem je takto namodelovaný proces zpracovat a zrealizovat v informačních systémech, kdy v tomto případě je třeba na celou operaci pohlížet z pohledu informatiky a tím zajistit správnost a korektnost. Nutností je, co nejpřesněji specifikovat vlastnosti a definovat korektní model procesu, následně je třeba tyto vlastnosti ověřit, případně odhalit či opravit. Níže uvedené vlastnosti jsou považovány jako základní požadavky pro následné využití modelovaných business procesů. Pomocí nejrozšířenějších metod UML a EPC je velmi náročné dodržení takto stanovených požadavků a takřka neproveditelné pomocí nástrojů výpočetní techniky a to hlavně z důvodu neexistence formální a přesné specifikace.

Mezi základní vlastnosti patří:

- dosažitelnost *– v procesu se nesmí nacházet aktivita (či jiný element), kterou není možné nikdy dosáhnout, tj. neexistuje takový řídící tok, který by vedl k realizaci daného elementu*
- živost *– jedná se o takový proces, který ve svém řídícím toku neobsahuje žádný deadlock (místo, kdy není možné realizovat další element) nebo lifelock (zacyklení bez možnosti opuštění nekonečné smyčky)*
- bezpečnost *– proces, ve kterém nemůže dojít k nekorektnímu provedení daného elementu (např. provedení aktivity, když ještě není možné aktivitu provést = chybějící podmínky v řídícím toku)*
- spolehlivost *– proces, jehož všechny elementy jsou dosažitelné z počátečního stavu a z těchto elementů je dosažitelný koncový stav a po ukončení procesu není žádný element (stav) aktivní*

(Převzato z www.uhk.panrepa.org/bpe_final/Kovarik_prace.doc)

2.1.9 Programovací jazyky a nástroje pro BPM

BPM Suite poskytuje programovací rozhraní (webové služby, rozhraní pro aplikační programy (API)), které umožňují podnikové aplikace, které mají využít BPM.

Programovací jazyky, které jsou zavedeny pro BPM jsou:

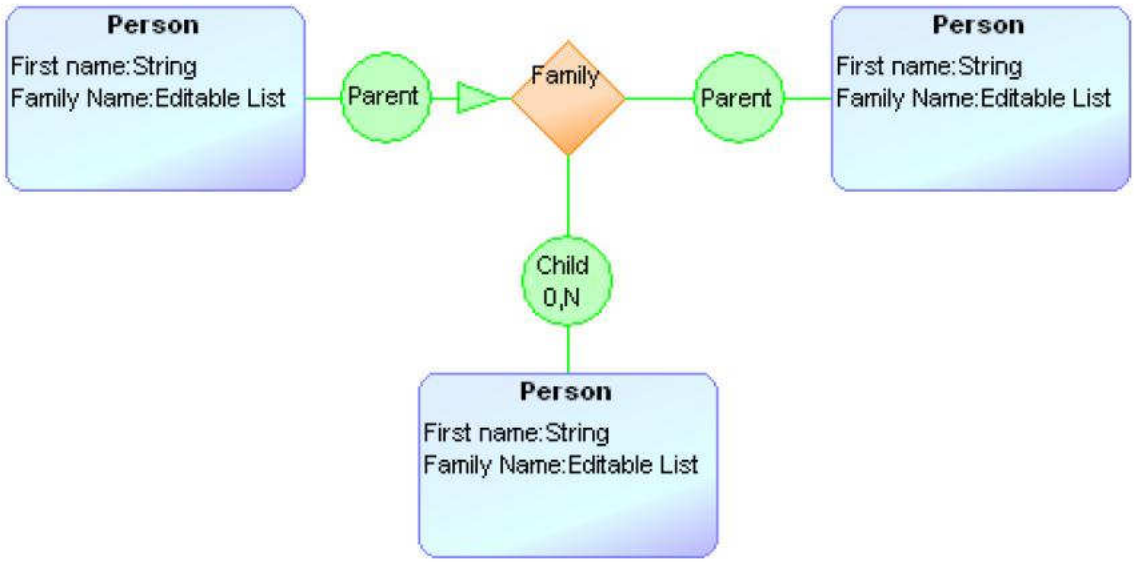
- Architecture of Integrated Information Systems (ARIS) podporující EPC
- Business Process Execution Language (BPEL)
- Web Services Choreography Description Language (WS-CDL)
- XML Process Definition Language (XPDL)
- Java Process Definition Language (JBPM)

2.2 Metamodelování

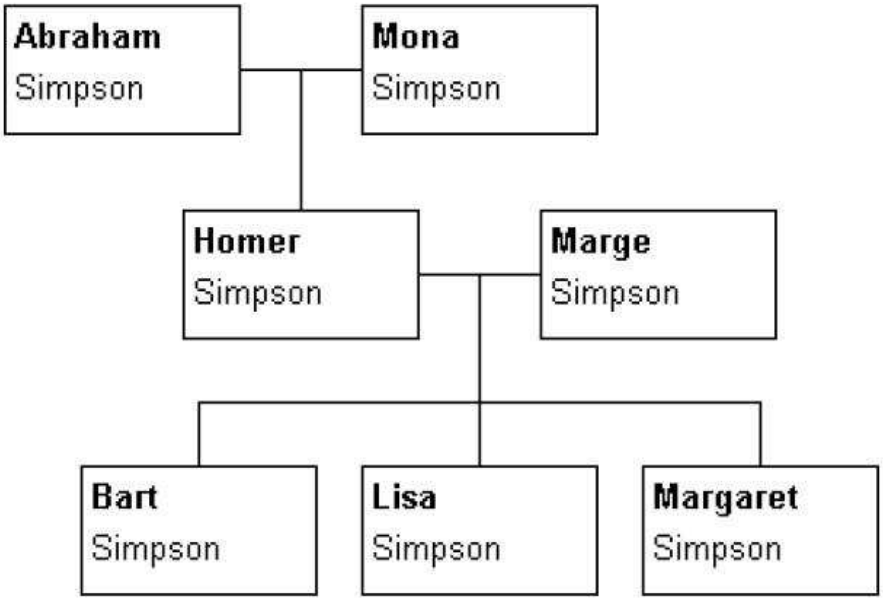
Metamodelování se zabývá modelováním tzv. meta-modelů. Ale ještě než se začneme zabývat samotným meta-modelováním, je důležité vysvětlit význam tohoto slova pro lepší pochopení celé problematiky. Předpona „meta“ znamená přidání nadřazené a zobecněné vrstvy abstrakce, neboli také jinak vysvětleno jako odpoutání se od určité problematiky a zaměření se na celý problém v širším, globálním úhlu pohledu, který by měl vést k jistému zobecnění celého přístupu v dané problematice. Stručně řečeno v terminologii informatiky je prefix „meta“ přeložen jako výraz pro zobecnění. Význam slova model je vysvětlován jako popis skutečnosti reálného světa, při které se v maximální míře odbourají informace, které jsou nepotřebné pro vypovídající vlastnost modelu a současně je nutné dostatečným způsobem zachytit stávající stav dané skutečnosti. Tudíž metamodel můžeme chápat jako popis struktury a vlastností daného modelu, neboli také schéma modelu. V praxi metamodel vychází z popisu modelu jako systému, používaného pro vývoj aplikací, informačních systému a simulací výše zmíněných business procesů.

Například můžeme říct, že metajazyk je jazyk, který popisuje jazyky. Konkrétněji se na celou problematiku můžeme podívat třeba tak, že například metadata můžou představovat schéma relační databáze a data skutečné hodnoty v tabulkách, analogicky přeneseno je možné říci, že metamodel je model popisující běžné modely.

Níže uvedené obrázky (obrázek 5 a obrázek 6) znázorňují metamodel rodinného stromu a následně příklad konkrétního modelu rodinného stromu, již vychází z daného metamodelu.



Obrázek 4 - Metamodel rodinného stromu



Obrázek 5 - Konkrétní model rodinného stromu

Důvod, který vedl ke vzniku metamodelování je ten, že je potřeba popsat odlišné datové struktury, které jsou používány jako komponenty navrhovaných modelů.

Metamodely se běžně používají jako:

- *schéma pro sémantická data, která je potřeba uložit nebo předat mezi subjekty*
- *jazyk podporující konkrétní schéma nebo proces*
- *jazyk vyjadřující doplňující sémantiku přidanou k již existujícím informacím*

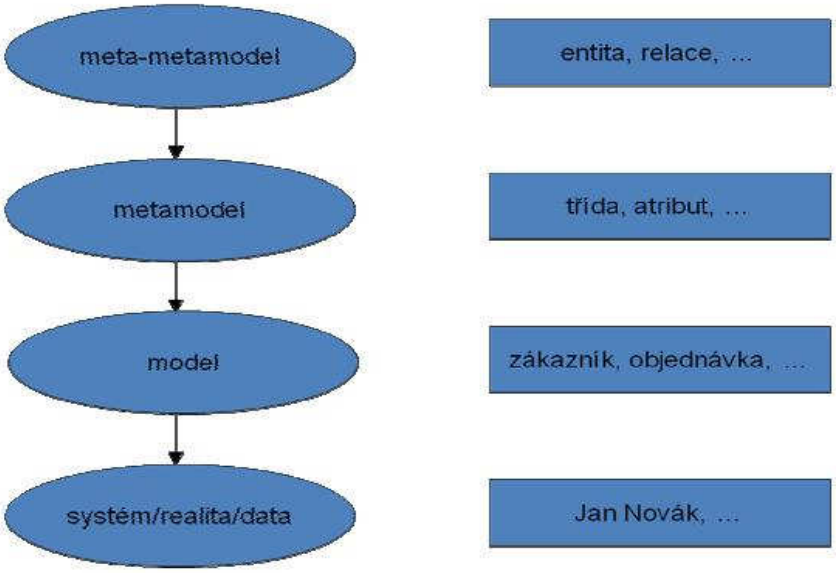
(Převzato z www.panrepa.org/CASE/zima2007/meta_case_zima2007.pdf)

2.2.1 Meta-metamodelování

Pokud jsme byli schopni popsat realitu skutečného světa pomocí modelu a umíme-li definovat další nadřazenou vrstvu abstrakce nad tímto modelem nazvanou metamodel, lze tedy z logického hlediska dojít k myšlence, že jsme schopni se na tento metamodel dívat z další nadřazené vrstvy, což znamená, že je možné pohlížet na tento metamodel jako na strukturu, jejíchž komponenty lze opět generalizovat a tím vytvořit jakousi vyšší úroveň náhledu. Tuto vyšší úroveň nazýváme meta-metamodel. Tento postup zobecnění lze neomezeně opakovat do nekonečna, ale v praxi se využívá maximálně čtvrtá úroveň abstrakce. Vede k tomu důvod, že s každou další vrstvou abstrakce se nám ztrácí adekvátnost činnosti a problém se stane natolik abstraktní, že ztratí vypovídající hodnotu pro návrh systému. Tudíž v této oblasti můžeme spíše teoretizovat.

Meta-metamodel většinou obsahuje informace o objektech, se kterými můžeme dále pracovat v nižších vrstvách, jako je metamodel či model. Metamodel specifikuje využitelnost těchto objektů.

Pro práci s metamodely je důležité znát meta-metamodel, ze kterého metamodel vychází. Např.: metamodel diagramu tříd v UML je postaven na nejpoužívanějším meta-metamodelu MOF . MOF popisuje, že metamodely na něm postavené obsahují třídy, objekty, vztahy atd. Mezi další meta-metamodely patří COMMA či GOPPR.



Obrázek 6 - Vrstvy metamodelování

2.2.2 COMMA

Projekt COMMA (Common meta model architecture) popisuje metamodel na bázi pojmů ze světa objektově orientovaných metodologií. COMMA používá tyto základní pojmy:

***- pojem (Concept)** – má jméno a atributy*

***- dědění (Inheritance)** – vyjadřuje relaci specializace*

***- asociace (Association)** – vyjadřuje vztah mezi pojmy*

***- agregace (Aggregation)** – vyjadřuje skládání, je to speciální případ asociace*

***- role (Role)** – objevuje se, když objekt přijímá charakteristiky jiného objektu. Role je dočasná a objekt může mít i více rolí najednou.*

(Převzato z www.agris.cz/etc/textforwarder.php?iType=2&iId=137547)

2.2.3 GOPRR

Základní prvky GOPRR (graph-object-property-role-relationship) jsou:

***- diagram (Graph)** – je kolekce objektů, vztahů a rolí, která definuje, co a jak lze spojovat dohromady*

***- objekt (Object)** – definuje entitu, která může existovat sama o sobě*

***- vlastnost (Property)** – charakterizuje graf, objekt, roli nebo vztah*

- role (Role) *– existuje mezi vztahem a objektem*

- vztah (Relationship) *– existuje mezi dvěma a více objekty*

(Převzato z www.agris.cz/etc/textforwarder.php?iType=2&iId=137547)

2.2.4 MOF

MOF (Meta Object Facility) je standardem konsorcia *OMG (Object Management Group)* pro modelově řízené inženýrství.

MOF jako jazyk:

- *specifikuje, vytváří a upravuje například práci s technologií neurálních metamodelů*
- *zajišťuje konzistenci mezi jednotlivými sadami modelů*
- *implementuje design a použití repositoří aplikačních vývojových nástrojů*

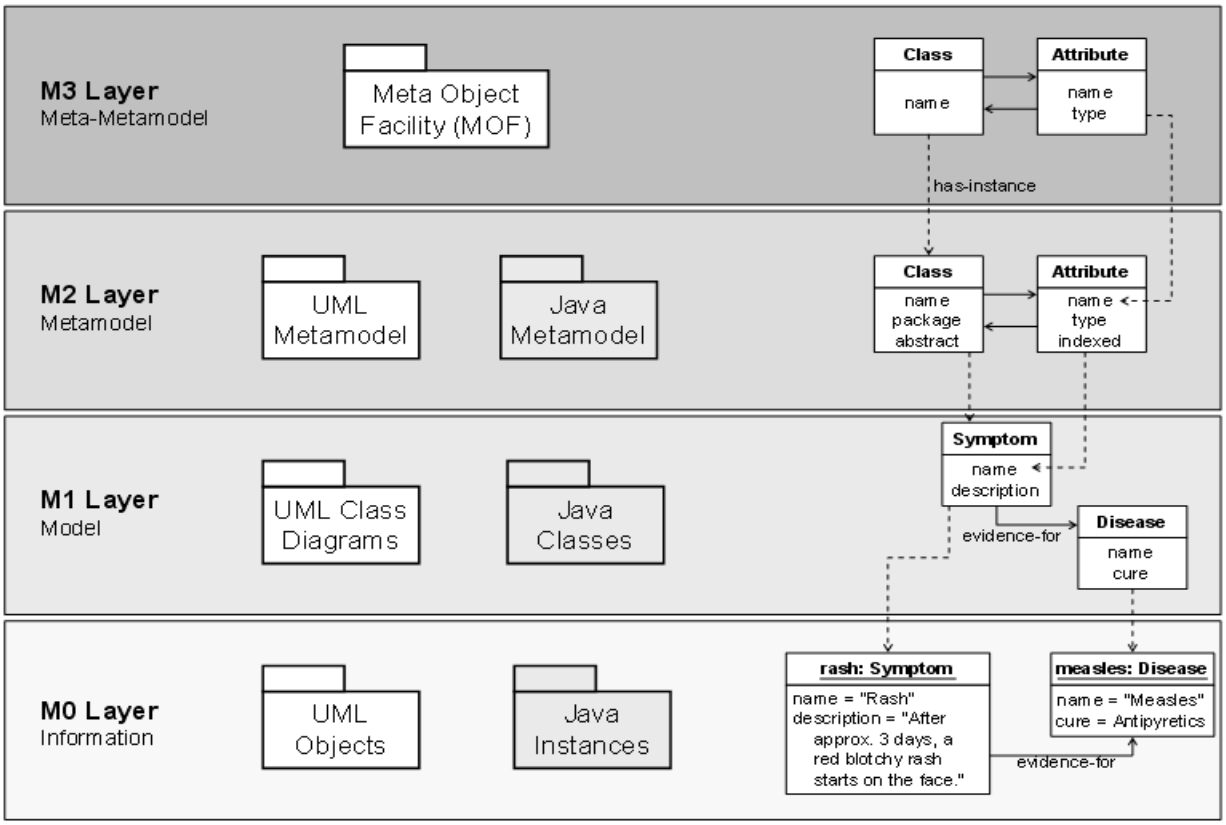
Repositoř se tedy stává úložištěm metadat (resp. jimi popsaných modelů), která za pomocí pravidel definovaných obdobnými metadaty hlídá svou integritu, resp. integritu modelů popsaných metadaty v ní obsažené.

MOF je objektově orientovaný jazyk, který popisuje sám sebe, díky čemuž je vysoce modifikovatelný a rozšiřitelný. Současně definuje komunikační rozhraní, používaná pro správu a operace s modely, jejichž rozhraní dosud nebyla použita; k tomu používá Interface Definition Language (IDL) CORBA. Typickými metamodely (metamodelovacími jazyky) navrženými OMG jsou UML, Sysel, SPEM nebo CWM

.(Převzato z www.panrepa.org/CASE/zima2007/meta_case_zima2007.pdf)

Jak možno vyčíst z obrázku obr.: 7 MOF je složen ze čtyř vrstev, z nichž vrstva vyšší je meta-vrstvou k vrstvě nižší.

- M3 MOF model
- M2 například představuje jazyk UML
- M1 konkrétní model
- M0 vygenerovaný kód z modelu

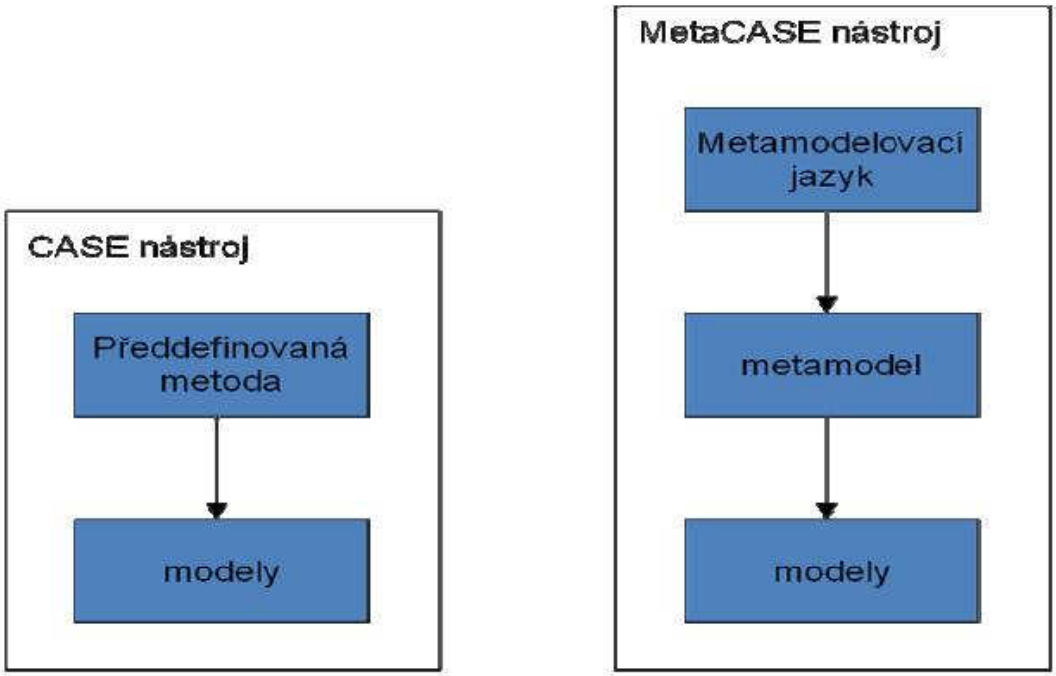


Obrázek 7 - Vrstvy MOF

2.3 Case a metaCase nástroje

Jak již bylo řečeno, metamodel je nadřazenou vrstvou od něj odvozené vrstvy modelů, tím pádem je velmi podstatný pro vývoj CASE nástrojů, které následně využíváme k usnadnění modelování těchto modelů. Existují také nástroje, které využívají metamodelovací přístup. Tyto nástroje nazýváme Meta-Case nástroje a oproti Case nástrojům mají schopnost úpravy jednotlivých metod, které používáme při definování metadat.

Závislost mezi Case a metaCase nástroji je obdobná jako mezi modelem a metamodelem, tudíž lze říct, že metaCase je jiný pohled na metamodelování, při kterém využíváme Case nástrojů s určitou úrovní abstrakce. Dle vybraného meta-metamodelu se také liší jednotlivé metaCase nástroje. Jak je zřejmé z obrázku obr.: 8, Case nástroje využívají pouze dvě nejnižší vrstvy oproti metaCase nástrojům, které využívají vrstvy tři a v případě pohledu z úrovně meta-metamodelu dokonce čtyři. Podle ISO jedním z jazyků popisujících metamodely je standard Meta Object Facility (MOF), definovaný skupinou Object Management Group (OMG).



Obrázek 8 - Porovnání vrstev Case a MetaCase

2.4 OMG

Object Management Group (OMG), je konsorcium, původně zaměřené na stanovení standardů pro distribuované objektově-orientované systémy. Nyní se zaměřuje na vytváření standardů pro modelování aplikací, systémů a podnikových procesů. OMG se ubírá směrem k vytváření standardů pro modelování tím, že vytváří standardy pro Unified Modeling Language (UML) a následně související standardy pro:

- Meta-Object Facility (MOF)
- XML Metadata Interchange (XMI)
- MOF Query/Views/Transformation (QVT)

2.5 XML metadata interchange (XMI)

Pro přenos metadat namodelovaných pomocí MOFu byl vyvinut konsorciem OMG standard XML metadata interchange (XMI). XMI je v podstatě jazyk na bázi XML, který byl vyvinut k popisu modelů na bázi MOFu. XMI soubor je textový soubor s XML formátováním dat generovaný z UML CASE nástrojů. XMI je mezinárodně uznávaným standardem podporovaným většinou CASE nástrojů.

Účelem XMI je umožnit snadnou výměnu metadat mezi UML-modelovacími nástroji a repositáři metadat založených na MOF v distribuovaných heterogenních prostředích. XMI integruje tři průmyslové standardy:

1. XML - eXtensible Markup Language, standard W3C
2. UML - Unified Modeling Language, modelovací OMG standard
3. MOF - Meta objekt Facility, meta-OMG modelování a metadata repository standard

Integrace těchto tří norem do XMI umožňuje vývojářům distribuovaných systémů sdílení objektů modelů a další metadata

2.6 Unified Modeling Language (UML)

Jedná se o grafický jazyk, který slouží k vizualizaci, specifikaci, navrhování a dokumentaci programových systémů v softwarovém inženýrství. UML nabízí standardní způsob zápisu pro:

- *návrh systému včetně konceptuálních prvků jako jsou business procesy a systémové funkce*
- *konkrétní prvky jako jsou příkazy programovacího jazyka, databázová schémata a znovupoužitelné programové komponenty*

(Převzato z cs.wikipedia.org/wiki/Unified_Modeling_Language)

UML podporuje objektově orientovaný přístup k analýze, návrhu a také popisu programových systémů ovšem neobsahuje způsob, jak by se měl používat, ani neobsahuje metodiku pro analýzu, specifikaci a návrh programových systémů. Standard UML definuje standardizační skupina Object Management Group (OMG).

2.6.1 Aktivitní Diagram

Diagramy aktivit představují obchodní a provozní workflow systému. Diagram aktivit je dynamický diagram, který ukazuje aktivity a událostí, které způsobí objekt. Diagram aktivit je jedním z UML diagramů, které popisují chování. Tento diagram se používá pro modelování procedurální logiky, procesů a zachycení workflow. Každý proces v diagramu aktivit je reprezentován sekvencí jednotlivých kroků, které jsou v modelu zakresleny jako:

- **akce** – atomické dále nedělitelné kroky
- **vnořené aktivity** – volání jiných procesů (aktivit), tyto aktivity mohou být reprezentovány dalším diagramem aktivit.

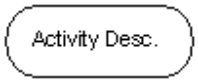
Sekvenci jednotlivých kroků v diagramu aktivit určuje řídicí tok

2.6.2 Aktivitní diagram - Grafická notace

Počáteční aktivita - ukazuje, výchozí bod nebo první aktivity toku, je značena plným kruhem

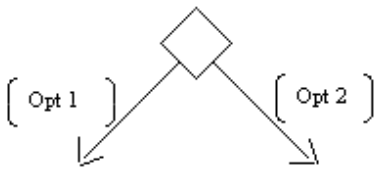
Obrázek 9 - Počáteční aktivita (UML- Aktivitní diagram)

Aktivita - zastoupena obdélníkem se zaoblenými hranami



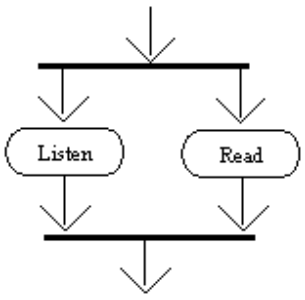
Obrázek 10 - Aktivita (UML - Aktivitní diagram)

Rozhodování - logika, kde rozhodování je znázorněno diamantem a šipkami na obou stranách vycházejících z diamantu



Obrázek 11 - Rozhodování (UML - Aktivitní diagram)

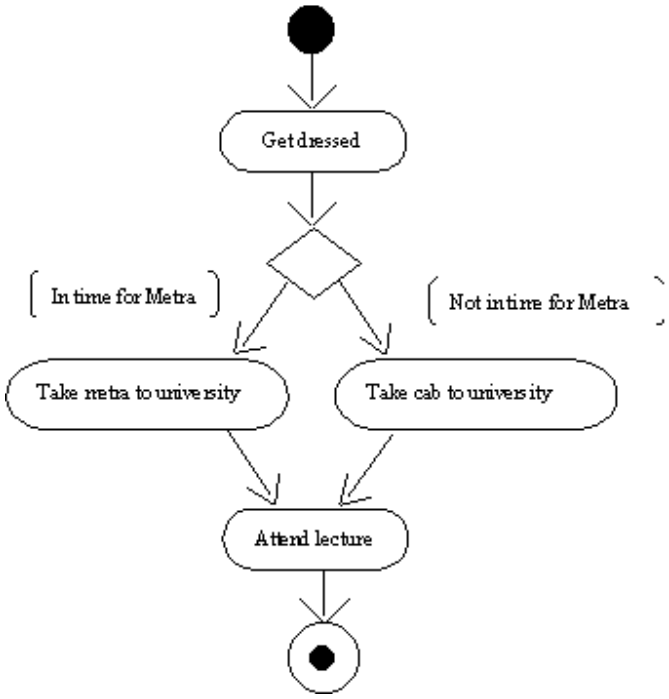
Souběžné činnosti - některé činnosti se vyskytují současně nebo paralelně. Tyto činnosti jsou tzv. souběžné činnosti. Například poslech přednášející při pohledu na tabuli je paralelní činnost. To je reprezentováno horizontálním splitem (husté tmavé linie), dále dvě souběžné činnosti vedle sebe a nakonec vodorovná čára opět ukazuje konec paralelní činnosti



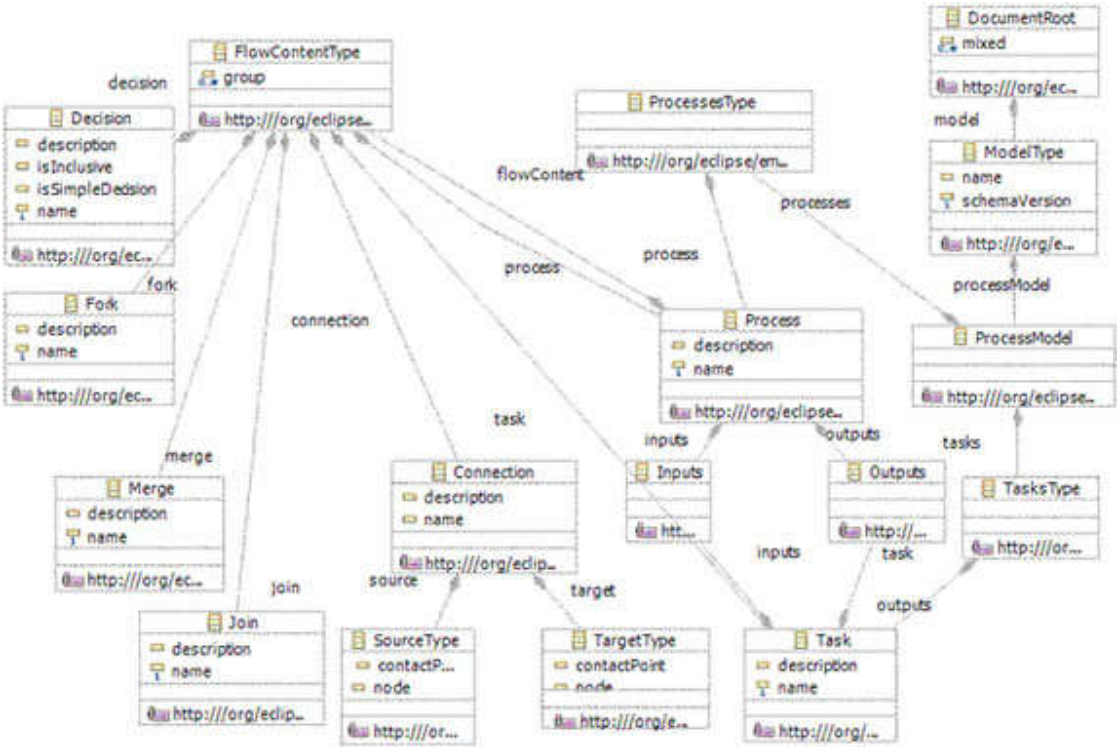
Obrázek 12 - Souběžné činnosti (UML - Aktivitní diagram)

Závěrečná Aktivita - diagram používá symbol kruhu s černým středem. Je také nazýván jako poslední činnost

Obrázek 13 - Závěrečná aktivita (UML - Aktivitní diagram)



Obrázek 14 - Příklad aktivitního diagramu (UML - Aktivitní diagram)



Obrázek 15 - Aktivitní diagram (UML) Metamodel

2.7 Business Process Management Initiative (BPMN)

Jedná se o grafickou notaci, která je také využívána k modelování business procesů. Konkrétněji jde o soubor grafických objektů, které jsou mezi sebou spojovány dle určených pravidel touto notací. Vznik BPMN má na svědomí BPMI (Business Process Management Initiative), kdy hlavním cílem bylo vytvořit notaci čitelnou všemi zúčastněnými aktéry, jako například business analytiky, vývojáři a analytiky monitorující procesy, kteří zasahují do životního cyklu procesu. V důsledku, že BPMN používá mnoho nástrojů, se stalo téměř standardem ve světě modelování business procesů. Tato notace má hlavně velký přínos v oblasti komunikace mezi návrhem a samotnou implementací procesu.

BPMI se snažilo představit notaci, která bude jednoduchá na pochopení a používání a zároveň nabídne modelování i komplexních business procesů. Dalším důležitým krokem byla definice převodu mezi návrhem procesu v BPMN a jeho implementací například v BPEL, BPML. BPMN osahuje definice převoditelnosti jednotlivých elementů a sekvencí těchto elementů do jazyka BPEL. Je třeba zdůraznit, že díky benevolenci modelování pomocí notace BPMN, není možné vygenerovat BPEL automaticky, ovšem je možné tuto operaci provést manuálně. Je sice pravdou, že existují nástroje, které touto funkcí automatického převodu disponují, ale na druhou stranu je třeba počítat s určitými omezeními již ve fázi modelování.

Právě v této době je možné využívat BPMN ve verzi 1.1. Tato verze byla představena v lednu roku 2008. Ovšem stále používanější verzí zůstává 1.0 z únoru roku 2006, která se od verze 1.1 mnoho neliší. Verze 2.0, od které se očekávají zásadní změny, spatřila světlo světa v podobě první beta verze v září 2009. Finální podoba je očekávána v červnu 2010.

Cílem BPMN 2.0 je pracovat s jedinou specifikací pro nový Business Process Model a Notaci, která definuje:

- notaci
- metamodel
- formát pro výměnu

Všechny tyto složky by měly stále reprezentovat značku BPMN, ovšem s upravenými jmény, které budou pro každou složku specifické.

2.7.1 Business Process Diagram (BPD) - Grafická notace

V BPMN modelujeme pomocí jediného diagramu, tzv. Business Process Diagram (BPD). Tento model obsahuje sít grafických objektů, ve většině případů aktivity a zobrazení toků informací mezi nimi. Mezi klady modelování v BPMN je možno pokládat dobrou přehlednost. K tomu přispívá dobrá odlišitelnost jednotlivých objektů. Grafická podoba jednotlivých objektů má jasně definované tvary, ovšem autor má možnost volby vlastních barev, např.: když je objekt využíván k různým účelům. Dokonce je možno v některých případech použít vlastní grafický objekt za podmínky, že se nebude překrývat s jiným již existujícím objektem. Takto nově vytvořený objekt by měl pouze upřednostňovat či poskytovat doplňující informace, tudíž by neměl ovlivňovat samotný tok procesu.

Business Process Diagram obsahuje čtyři základní druhy grafických elementů, jež se ještě dále dělí na další podtypy. Následuje výpis základních informací o jednotlivých typech grafických objektů:

Flow Objects (Tokové objekty)
- objekty, které souvisí s tokem informací v procesu

Event (Událost)
- značí se kroužkem,
- přímo ovlivňují tok procesu
- události, jimiž proces začne, skončí, či které nastanou v jeho průběhu

Obrázek 16 - Událost (BPMN – Business proces diagram)

Activity (Aktivita)
- obdélník se zaoblenými rohy
- znázorňuje činnost či práci
- může být buďto atomická (tzv. Task) nebo v sobě může obsahovat samostatný proces, pak se tato aktivita nazývá subprocesem

Obrázek 17 - Aktivita (BPMN – Business proces diagram)

Gateway (Brána)

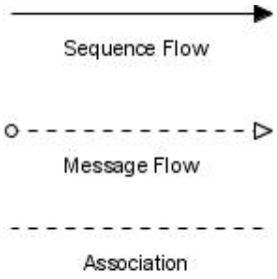
- značí se čtvercem či kosočtvercem, stojícím na špici
- označuje větvení či souběh toků procesu, např. rozhodování či paralelní zpracování

Obrázek 18 - Brána (BPMN – Business proces diagram)

Connecting Objects (Spojovací objekty)

- objekty, které slouží k spojení tokových objektů navzájem či s artefakty

- **Sequence Flow** (Sekvenční tok)
  - nepřerušovaná čára s vyplněnou šipkou
  - určuje sekvenci (pořadí) aktivit
- **Message Flow** (Tok zpráv)
  - přerušovaná čára s prázdnou šipkou
  - znázorňuje tok zpráv mezi dvěma účastníky procesu
- **Association** (Asociace)
  - přerušovaná čára
  - umožňuje spojit objekt s nějakou dodatečnou informací



Obrázek 19 - Spojovací objekty (BPMN – Business proces diagram)

Artifacts (Artefakty)

- značí nějaké upřesňující informace pro proces, nemají vliv na jeho tok

- **Data Object** (Datový objekt)
  - značí se obdélníkem s přehnutým rohem (list papíru),
  - reprezentuje data, se kterými pracují aktivity.
- **Group** (Seskupení)
  - obdélník kreslený přerušovanou čárou,
  - seskupení aktivit za analytických či dokumentačních důvodů.
- **Annotation** (Poznámka)
  - text, jenž je spojen asociací s jiným grafickým objektem,
  - poskytuje dodatečnou textovou informace.

Obrázek 20 - Artefakty (BPMN – Business proces diagram)

Swimlanes (Plavecké dráhy)

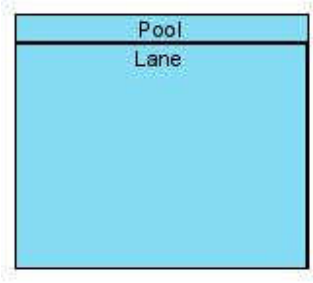
- slouží k zobrazení účastníků procesu či uspořádání činnosti v procesu např. podle rolí

Pool

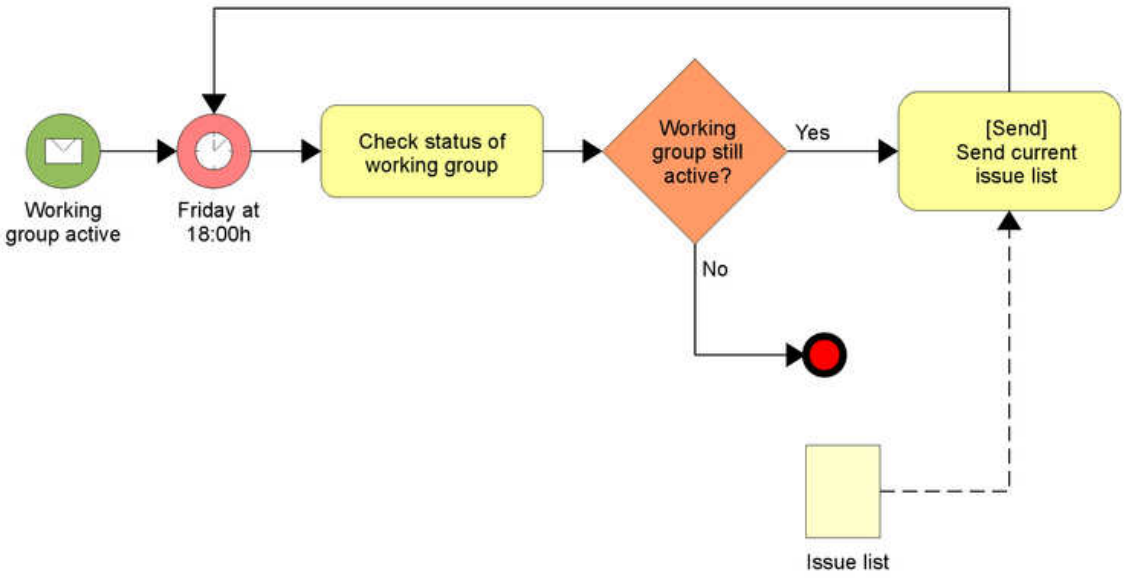
- ohraničuje proces, v jeho záhlaví je název poolu
- reprezentuje účastníka procesu
- v rámci jednoho poolu se nachází právě jeden samostatný process
- komunikace mezi pooly probíhá pomocí zpráv (message flow)

Lane (Dráha)

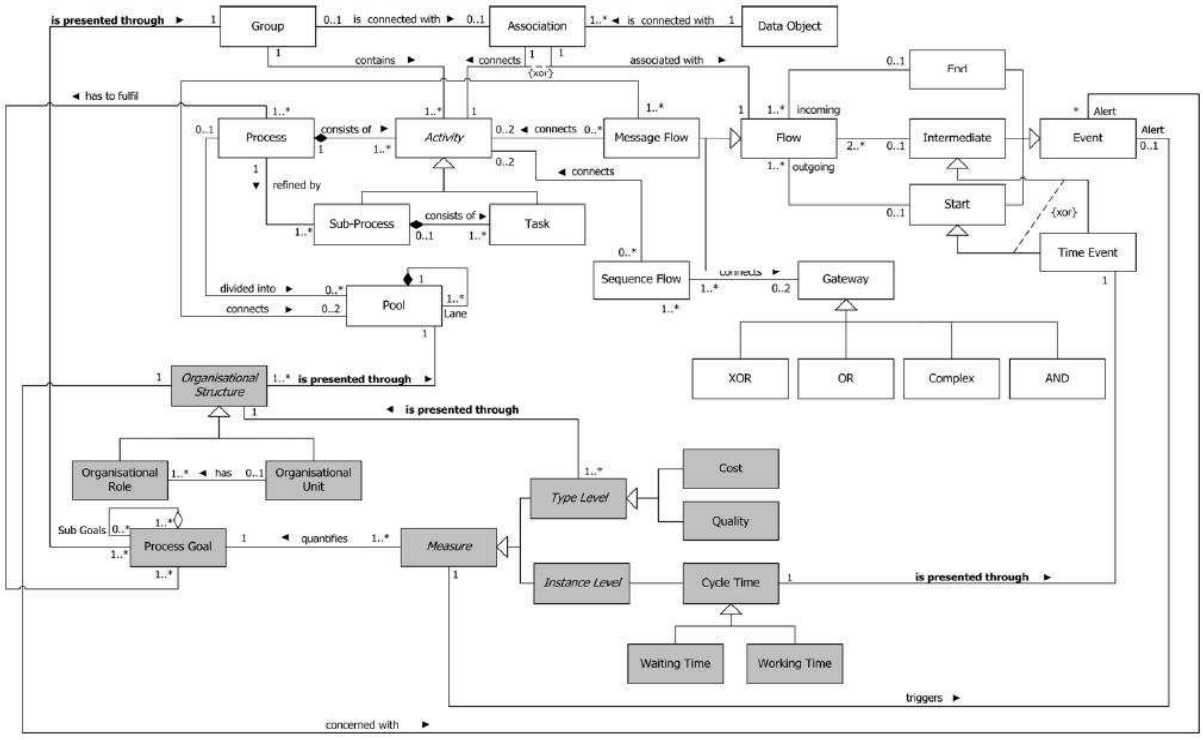
- podčást poolu
- slouží k uspořádání a kategorizaci aktivit
- může značit např. role, oddělení či funkce organizace
- komunikace mezi dráhami probíhá pomocí sekvenčního toku (sequence flow)



Obrázek 21 - Plavecké dráhy (BPMN – Business proces diagram)



Obrázek 22 - Příklad BPMN (BPMN – Business proces diagram)



Obrázek 23 - Extended BPMN Metamodel

2.8 Event – Driven Process Chain (EPC)

Jedná se o metodu, která rovněž patří k velmi rozšířeným metodám, a to hlavně z důvodu, že je nasazena a podporována velkými softwarovými systémy jako jsou SAP R/3 a ARIS, ale také díky své jednoduchosti ve spojování událostí a aktivit. Systém modelování pomocí EPC je použitelný i při vytváření relativně velmi složitých procesů. Tato metoda se používá k popisu procesu z pohledu návaznosti jednotlivých aktivit, časových posloupností aktivit či paralelismů.

2.8.1 EPC – Grafická notace

Aplikace využívající metodu EPC, je založena na třech základních prvcích, které pomocí zřetězení vytváří posloupnost aktivit, které jsou zainteresované do daného procesu.

Základní stavební prvky tvoří:

- **Aktivity (Activity, Function)**
  - pomocí aktivit je určeno, co má být v danou chvíli v rámci procesu vykonáno.
  - aktivity v EPC diagramech mohou mít právě jeden vstup a jeden výstup

Obrázek 24 - Aktivita (EPC)

- **Události (Event)**
  - události popisují situace před či po vykonání dané aktivity
  - tvoří jakýsi propojovací bod mezi jednotlivými aktivitami, tzn. že výstupní událost jedné aktivity může tvořit vstupní událost (podmínku) následné aktivity

Obrázek 25 - Událost (EPC)

- **Logické spojky (Connectors)**
  - konektory slouží ke spojování událostí a aktivit
  - tento prvek je nezbytný pro popis řídícího toku procesu, kdy jej mohou rozdělovat nebo naopak slučovat
  - v metodě EPC se používají tří typy spojek, a to AND, OR a XOR.

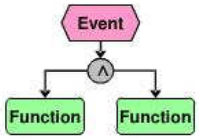
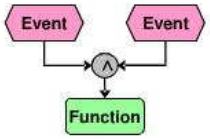
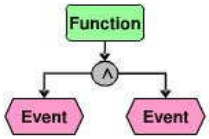
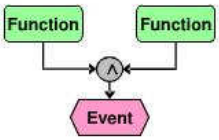
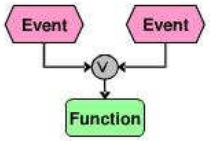
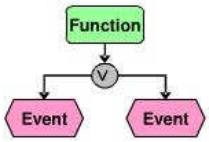
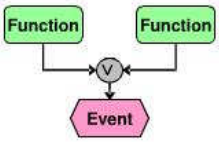
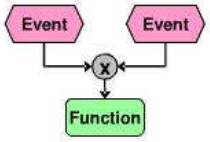
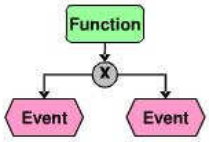
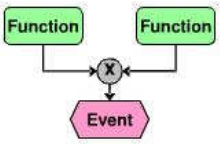
Obrázek 26 - Logické spojky (EPC)

- **XOR** (větvení a sloučení) – rozhodování, jakou cestu zvolit mezi několika kontrolními toky. Větvení může mít jeden vstupní kontrolní tok a dva nebo více výstupních. Pokud je splněna podmínka, větvení aktivuje právě jen jeden z odcházejících kontrolních toků a deaktivuje ostatní. Protějškem větvení je sloučení. Sloučení může mít dva nebo více příchozích toků a jeden výstupní kontrolní tok. Kontrola je předána dalšímu prvku po sloučení. Větvení je v EPC je reprezentováno jako otevírací XOR a sloučení je reprezentováno jako uzavírací XOR.
- **AND** - větvení a spojení aktivuje všechny cesty kontrolních toků souběžně. Větvení může mít jeden vstupní kontrolní tok a dva nebo více výstupních. Pokud je splněna podmínka, větvení aktivuje všechny výstupní kontrolní toky paralelně. Větvení v EPC je reprezentováno otevíracím AND a následné sloučení je reprezentováno jako uzavírací konektor AND.
- **OR** - aktivuje jednu nebo více cest mimo kontrolní tok. Otevírací OR konektor může mít jeden vstupní kontrolní tok a dva nebo více výstupní. Je-li podmínka splněna, otevírací konektor OR aktivuje jeden nebo více kontrolních toků a deaktivuje ostatní. Protějškem je uzavírací OR. Pokud je alespoň jeden z příchozích kontrolních toků aktivován, uzavírací OR předá řízení dalšímu následujícímu prvku.

Při modelování EPC diagram je třeba dodržovat následující pravidla:

- EPC diagram musí začínat a končit událostí
- Aktivity a události musí být propojeny tak, aby každá událost byla vstupem či výstupem aktivity
- Aktivita nebo událost mohou mít pouze jeden vstupní a jeden výstupní logický operátor pro větvení i spojení

- Událost je pasivní element, který nemůže být použit jako vstup do logického operátoru OR split nebo XOR split (viz tabulka 1).

	Event Trigger		Function Trigger	
	Single	Multiple	Single	Multiple
AND				
OR	Not Allowed			
XOR	Not Allowed			

Tabulka 1 - Přehled logických spojek notace EPC

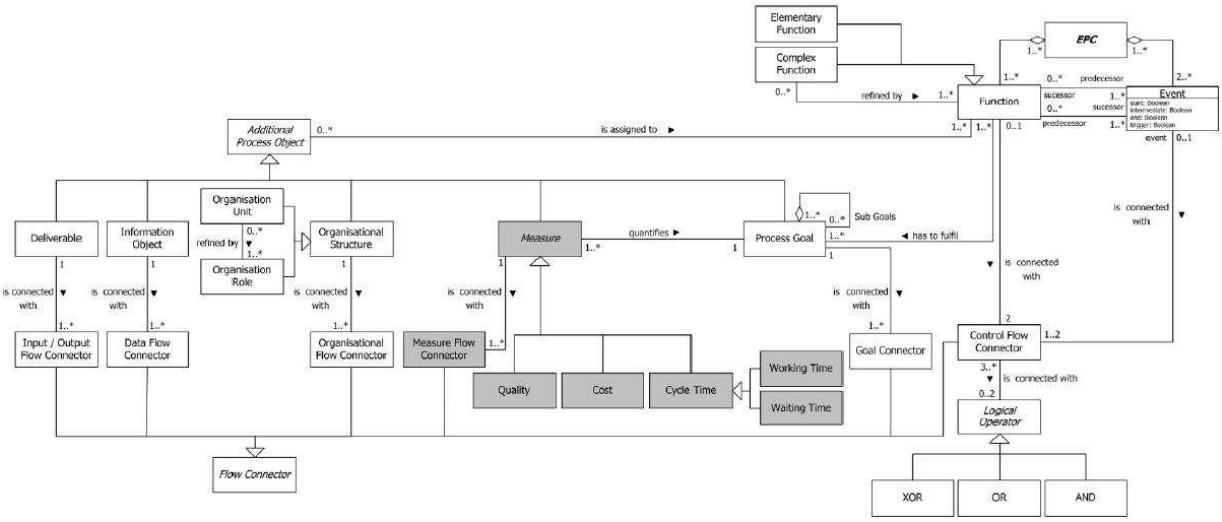
Kromě aktivit, událostí a logických spojek, které EPC využívá pro realizaci popisu business procesů, existuji i další rozšiřující grafické elementy, pomocí kterých jsme schopni popisovaný business proces blíže specifikovat. Většinou jde o prvky, které nám napomáhají vytvářet strukturu popisu procesu, jako například:

- vnořený proces
- rozhraní procesu
- odkaz na jiný proces

Pokud pracujeme s tzv. extended EPC, jsme schopni využívat další prvky, které přímo nesouvisející s průběhem vlastního procesu, jako například:

- externí požadavky dané aktivity
- zodpovědnosti za vykonávání aktivit

EPC má i svá negativa a to například, a to je třeba zdůraznit, že tato metodika dodnes nebyla zcela jasně formálně definována, což logicky znamená celou řadu problémů, například nejednoznačnost ve specifikaci procesů nebo zhoršená přenositelnost mezi jednotlivými softwarovými produkty. V případě realizace namodelovaného procesu pomocí přístupu ERP či WFM může dojít například k zacyklení či uvíznutí při jeho vykonávání.



Obrázek 27 - eEPC Metamodel

2.9 Petriho sítě

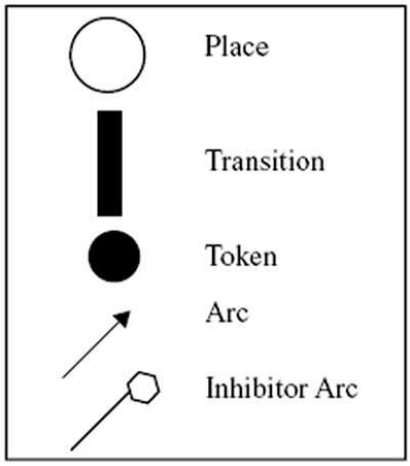
Petriho sítě byly vyvinuty jako přísně formalizovaná metoda, která měla za úkol rozšířit možnosti modelování konečných automatů. Pro svou přísnou formalizaci se značně odlišuje od výše uvedených metod business modelování. Tato metoda nabízí jednak velmi názorné grafické vyjádření, a také velmi solidní matematický aparát. Tato vlastnost je vítána především při realizaci a ověřování procesů. Jak už jsme zvyklí z předchozích notací i Petriho sítě nabízí několik základních modelovacích prvků. Princip modelování pomocí této metody je založen vytváření přechodů mezi jednotlivými místy, což je závislé na rozmístění tzv. tokenů v daných místech celé sítě.

2.9.1 Petriho sítě - Grafická notace

Hlavní modelovací elementy*:*

- ***Místo*** *(place) – místo je možné chápat např. jako událost v EPC diagramu a může obsahovat libovolný počet tokenů*
- ***Přechod*** *(transitiv) – jedná se o prvek analogický s aktivitou v EPC či diagramu aktivit a symbolizuje provádění dané činnosti*
- ***Token*** *– jedná se o element, který slouží k modelování vlastního průběhu řídícího toku procesu a symbolizuje aktuální stav celého procesu*

(Převzato z www.uhk.panrepa.org/bpe_final/Kovarik_prace.doc)



Obrázek 28 - Elementy (Petriho sítě)

Mimo tyto základní prvky, které jsou používané ve standardních Petriho sítích, existuje opět celá řada dalších rozšiřujících prvků a vlastností.

- *barvená PS umožňuje jednotlivé tokeny barevně odlišit a tím pádem následné přechody nejsou závislé nejen na vlastní existenci tokenů, ale také na jejich typu*
- *definované přechody umožňují realizovat složitější operace (AND, OR, atd.)*
- *zavedení časového aspektu*
- *strukturalizace a hierarchizace Petriho sítí*

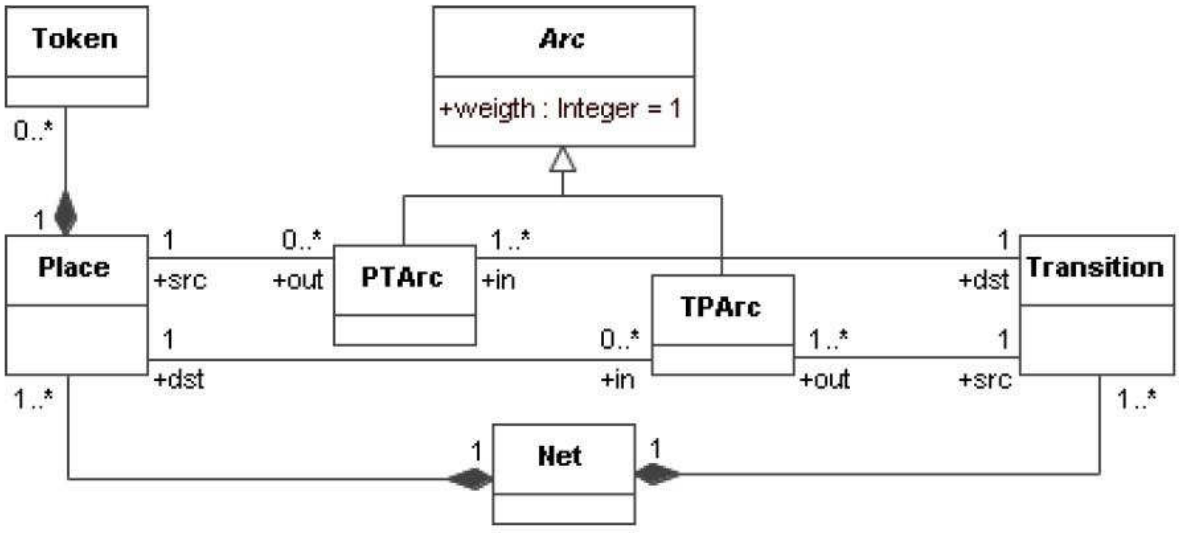
(Převzato z www.uhk.panrepa.org/bpe_final/Kovarik_prace.doc)

Existuje také modifikace Petriho sítí pro potřeby modelování business procesů, kterou označujeme jako WF-sítě (**WorkFlow** Nets). WF-sítě dodržují přísnou formalizaci Petriho sítí jako takových a navíc umožňují namodelovat specifické požadavky, které vyžaduje model business procesu, jako například:

- počáteční a koncové místo
- spouštění aktivity

I když se může zdát, že metoda Petriho sítě díky své přísné formalizaci a grafickému znázornění je nejlepší volbou pro modelování business procesů, není tomu tak zcela vždy. Předpokladem pro modelování pomocí této metody je, že autor ovládá a umí použít přinejmenším elementární znalosti Petriho sítě, čímž se stěží docílí jen pomocí intuitivního myšlení. V tomhle ohledu mají navrch předešlé metody, jejichž modely jsou čitelné i pro člověka, který je předtím nestudoval. V praxi je totiž důležité, aby se v modelech zorientovali i lidé, kteří neznají dané modelovací prostředí a nemají čas se učit formální definice Petriho sítí.

Na druhou stranu striktní formalizace a její relace na grafický jazyk je velkou výhodou Petriho sítí, protože tato kombinace je velmi dobře využitelná pro tvorbu simulací a ověřování business procesů. V poslední době se jako úplně nejvhodnější jeví kombinace Petriho sítí s metodou UML.

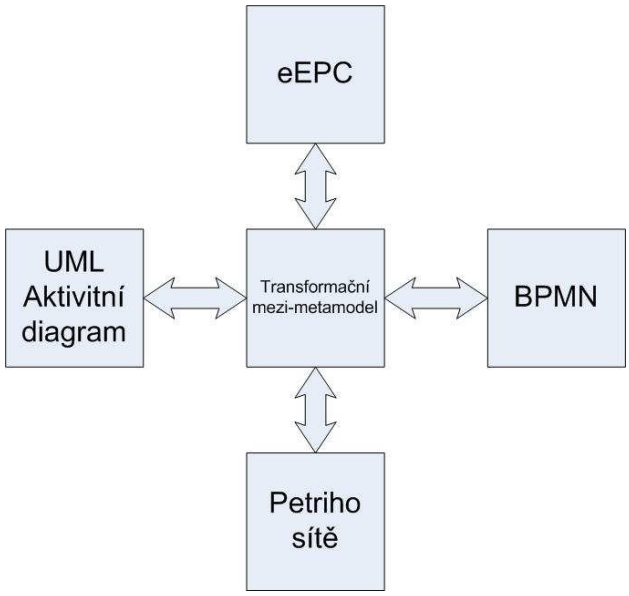


Obrázek 29 - Petriho sítě Metamodel

3 Praktická Část

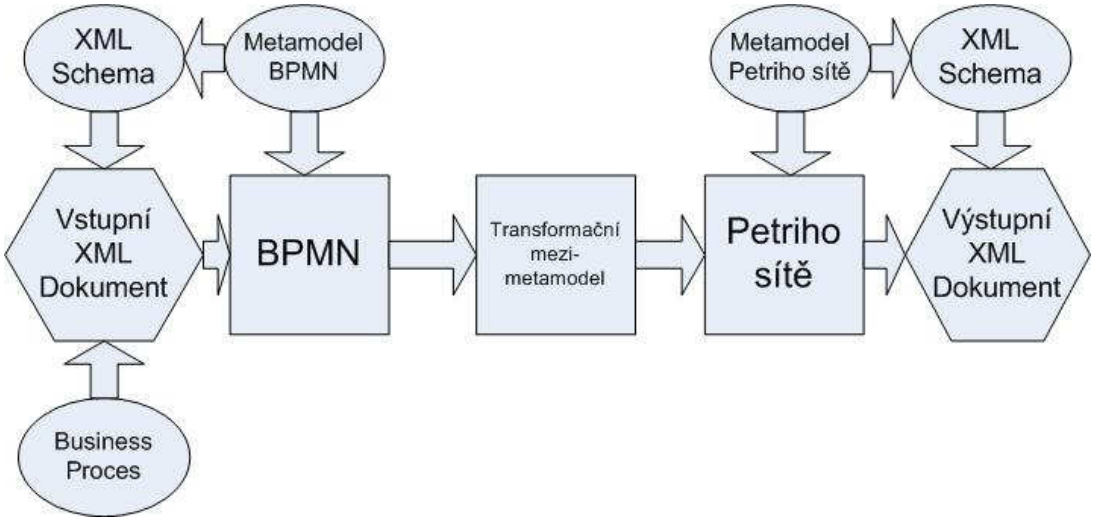
3.1 Úvod k praktické části

Cílem praktické části je vývoj aplikace, která bude schopna zrealizovat převody z jedné notace na druhou. Prvním krokem bude vytvoření metamodelů k jednotlivým notacím a navržení speciálního transformačního mezi-metamodelu, který bude obsahovat sjednocení všech elementů notací, které budeme využívat k převodu. Jednotlivé transformace se tedy budou realizovat přes tento metamodel. Důvod proč jsme tento metamodel zavedli, je vysvětlen v kapitole 3.2.1. Jak budou jednotlivé transformace probíhat, znázorňuje obrázek 30.



Obrázek 30 - Realizace transformací mezi notacemi

Uveďme si tedy příklad, že pokud budeme chtít převést BPMN na Petriho sítě, převedeme BPMN model na model, který vychází z mezi-metamodelu, a následně tento model převedeme na model Petriho sítí. Příklad je znázorněn na obrázku 31. Vzhledem k tomu, že mezi-metamodel bude obsahovat sjednocení všech elementů všech notací, které jsme definovali, je důležité říct, že první převod bude bezztrátový. Ovšem u převodu na výstupní notaci, může ke ztrátám dojít a to z důvodu, že výstupní notace nedefinuje všechny elementy jako notace vstupní.



Obrázek 31 - Příklad realizace transformace mezi BPMN a Petriho sítěmi

Dalším krokem bude nadefinování sad převodních pravidel k jednotlivým transformacím, podle kterých se proces transformace bude řídit. Jednotlivé převodní tabulky naleznete v kapitole 3.5.

Výsledná aplikace bude na vstupu číst model, který bude reprezentovat vstupní notaci, ve formátu XML dokumentu. Aby vstupní formát XML odpovídal námi namodelovanému metamodelu a měl jasně stanovenou strukturu, musíme převést jednotlivé metamodely vstupních notací na XML schéma, které jasně určuje strukturu XML dokumentu. Pokud tedy na vstupu budeme načítat XML dokument vycházející z daného XML Schéma, je zaručena korektnost načítaného dokumentu. Pro jasnější představu, jak by měl vstupní XML soubor vypadat, bude pro každou notaci vytvořen vzorový vstupní model, obsahující všechny námi definované elementy, jak v grafickém tak v XML formátu.

Posledním krokem v praktické části této práce je implementace samotné aplikace, která bude jednotlivé transformace realizovat. Tato aplikace na vstupu načte vstupní model v podobě XML dokumentu notace, ze které budeme transformaci realizovat, a následně ji přetransformuje na výstupní XML dokument námi zadané výstupní notace. Implementace bude realizována pomocí programovacího jazyku Java v prostředí Eclipse.

Na závěr provedeme ukázkovou transformaci, jejíž vstupní i výstupní XML dokumenty budou převedeny do grafické podoby pro jasnější demonstraci rozdílů mezi vstupním a výstupním modelem.

3.2 Metamodely

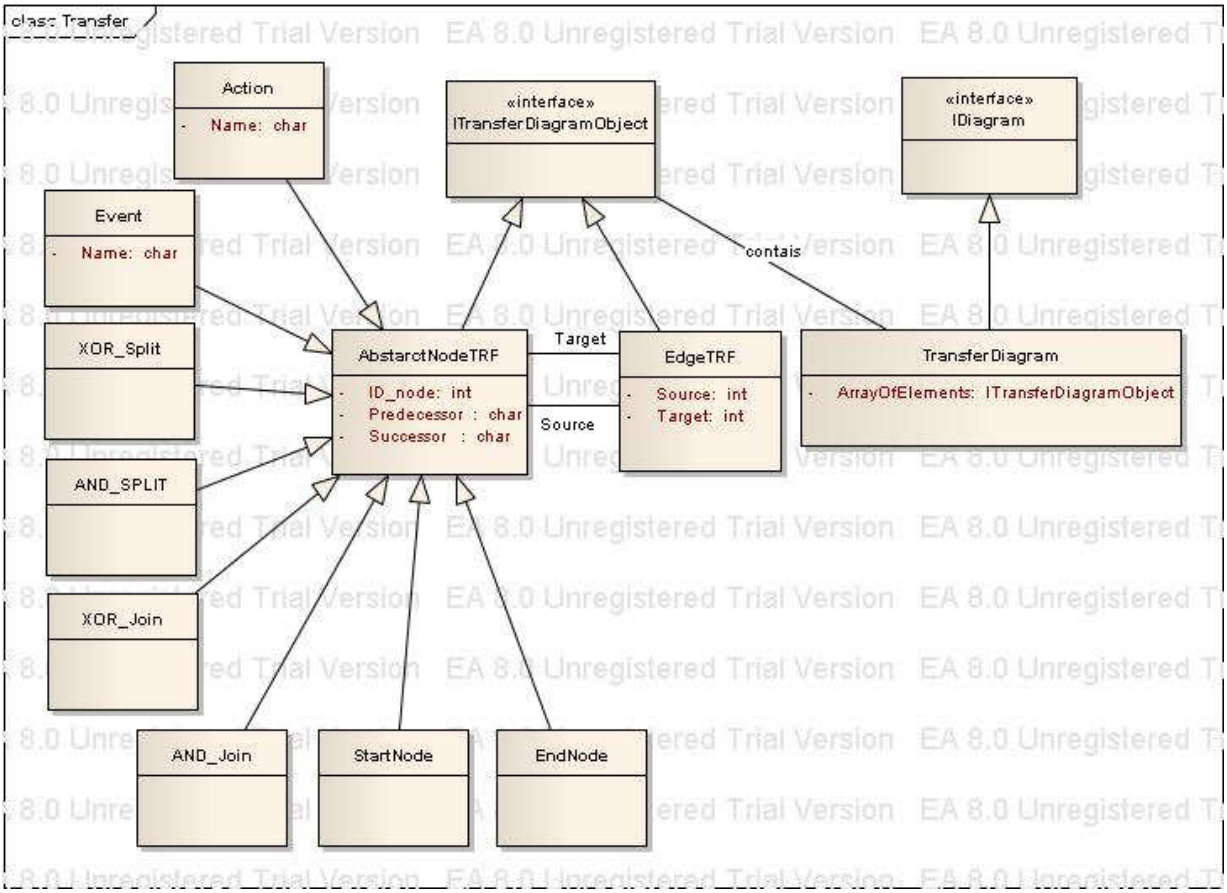
Vzhledem k tomu, že rostou potřeby uživatelů, kteří se zabývají modelováním business procesů, tím pádem se stále rozšiřuje počet elementů u notací, kterými je možné popsat chování reálného světa. Původní metamodely byly přepracovány z důvodu lepší synchronizace mezi metamodely jednotlivých notací, což se projeví v jasnější a čistší implementaci aplikace. Pro realizaci našich transformací jsme vytvořili metamodely, které zachycují nejdůležitější a nejpoužívanější elementy každé notace. Pro vizualizaci jednotlivých metamodelů jsme použili třídní diagramy notace UML.

3.2.1 Mezi-metamodel

Tento mezi-metamodel nám bude sloužit jako transformační prostředník, kdy každá notace bude převeditelná na tento metamodel a zpět. Mezi-metamodel jsme zavedli z důvodu nižšího počtu transformací. Našim záměrem je vytvořit transformace mezi všemi čtyřmi notacemi, což by znamenalo udělat transformace M:N, pokud zavedeme pomocný mezi-metamodel, budeme realizovat pouze počet transformací., který je roven počtu notací krát 2. Další výhodou bude případné rozšíření naší aplikace o další notace, kdy bude pouze nutné nadefinovat převodní pravidla mezi novou notací a transformační notací vycházející z transformačního mezi-metamodelu. V opačném případě bychom museli definovat pravidla mezi naší novou a všemi ostatními notacemi.

Pomocí mezi-metamodelu budeme schopni vytvářet modely, které budou složeny z následujících elementů.

- Aktivita
- Událost
- Sekvenční (kontrolní) tok, který spojuje jednotlivé elementy
- Startovací element
- Ukončovací element
- Element pro rozhodovací blok split - s podmínkami pro větvení (XOR split)
- Element pro rozhodovací blok join (XOR join)
- Paralelní tok split (AND split)
- Paralelní tok join (AND join)
- Logická spojka OR split
- Logická spojka OR join

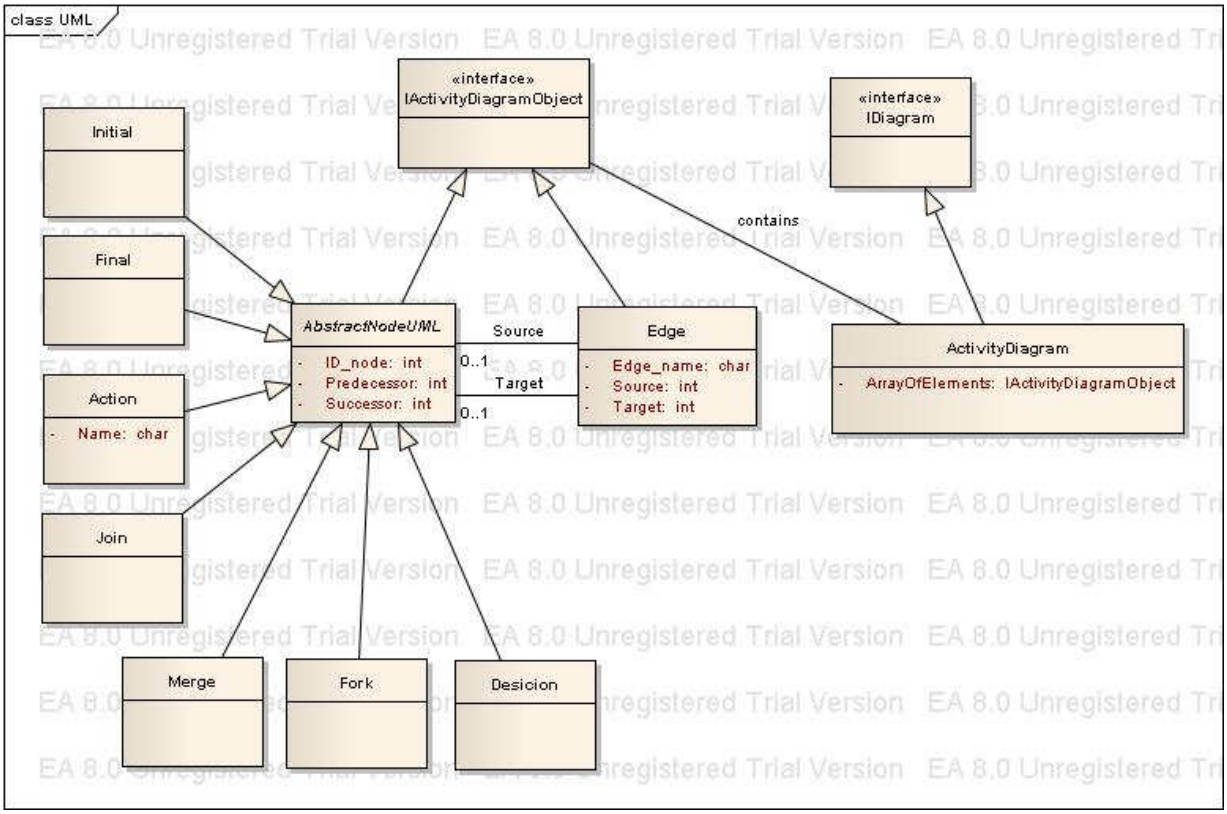


Obrázek 32 - Namodelovaný transformační mezi- metamodel

3.2.2 Metamodel UML – Aktivitní diagram

Pomocí vytvořeného metamodelu aktivitního diagramu budeme schopni vytvářet modely, které budou obsahovat následující elementy

- Počáteční stav
- Koncový stav
- Aktivita
- Strážní podmínka (v jiných notací je reprezentována událostí)
- Element pro rozhodovací blok typu split
- Element pro rozhodovací blok typu join
- Element pro synchronizaci (paralelní tok) blok typu join
- Element pro synchronizaci (paralelní tok) blok typu split
- Sekvenční (kontrolní) tok Edge, který spojuje jednotlivé elementy



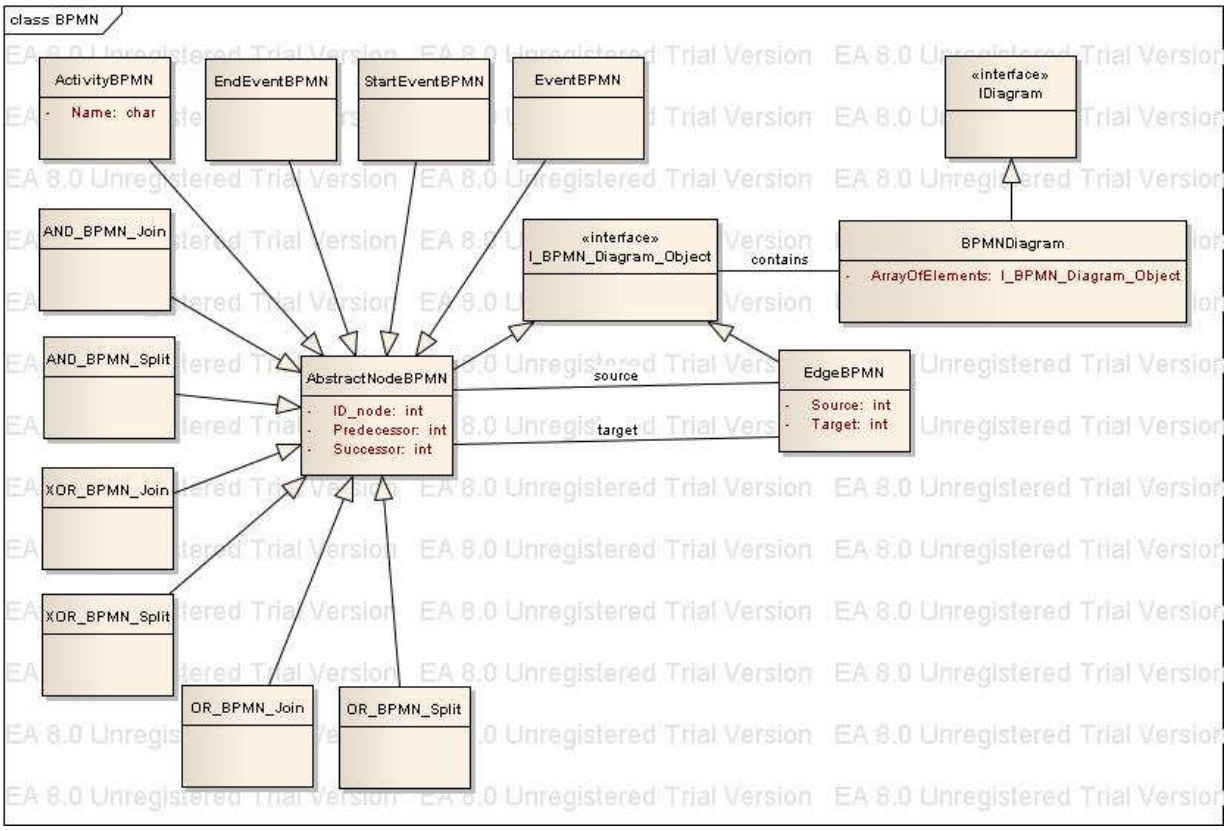
Obrázek 33 - Upravený Aktivitní Diagram (UML) Metamodel

3.2.3 Metamodel BPMN – Business Proces Diagram (BPD)

Pomocí vytvořeného metamodelu business proces diagramu budeme schopni vytvářet modely, které budou obsahovat následující elementy

- Počáteční událost
- Koncová událost
- Událost
- Aktivita
- Logická spojka AND typu split
- Logická spojka AND typu join

- Logická spojka XOR typu split
- Logická spojka XOR typu join
- Logická spojka OR typu split
- Logická spojka OR typu join
- Sekvenční (kontrolní) tok (Edge), který spojuje jednotlivé elementy



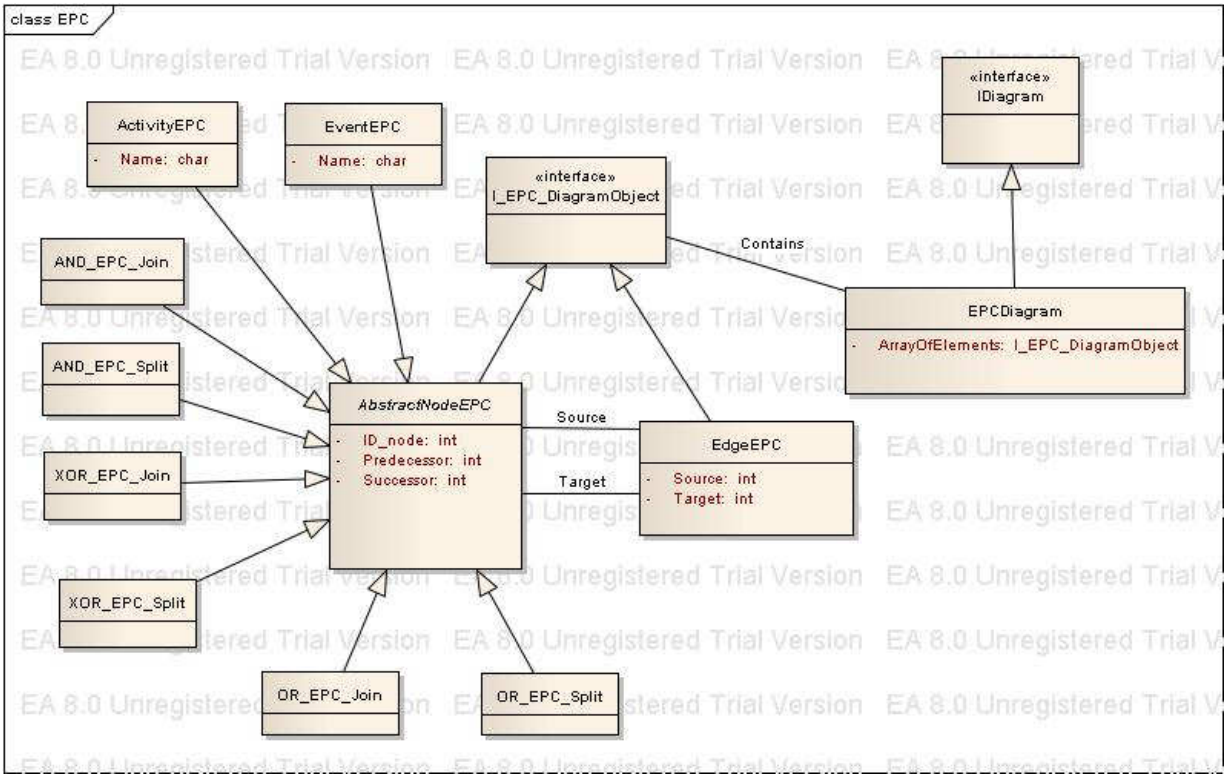
Obrázek 34 - Upravený BPMN Metamodel

3.2.4 Metamodel EPC

Pomocí vytvořeného metamodelu EPC diagramu budeme schopni vytvářet modely, které budou obsahovat následující elementy

- Událost
- Aktivita
- Logická spojka AND typu split
- Logická spojka AND typu join
- Logická spojka XOR typu split

- Logická spojka XOR typu join
- Logická spojka OR typu split
- Logická spojka OR typu join
- Sekvenční (kontrolní) tok (Edge), který spojuje jednotlivé elementy

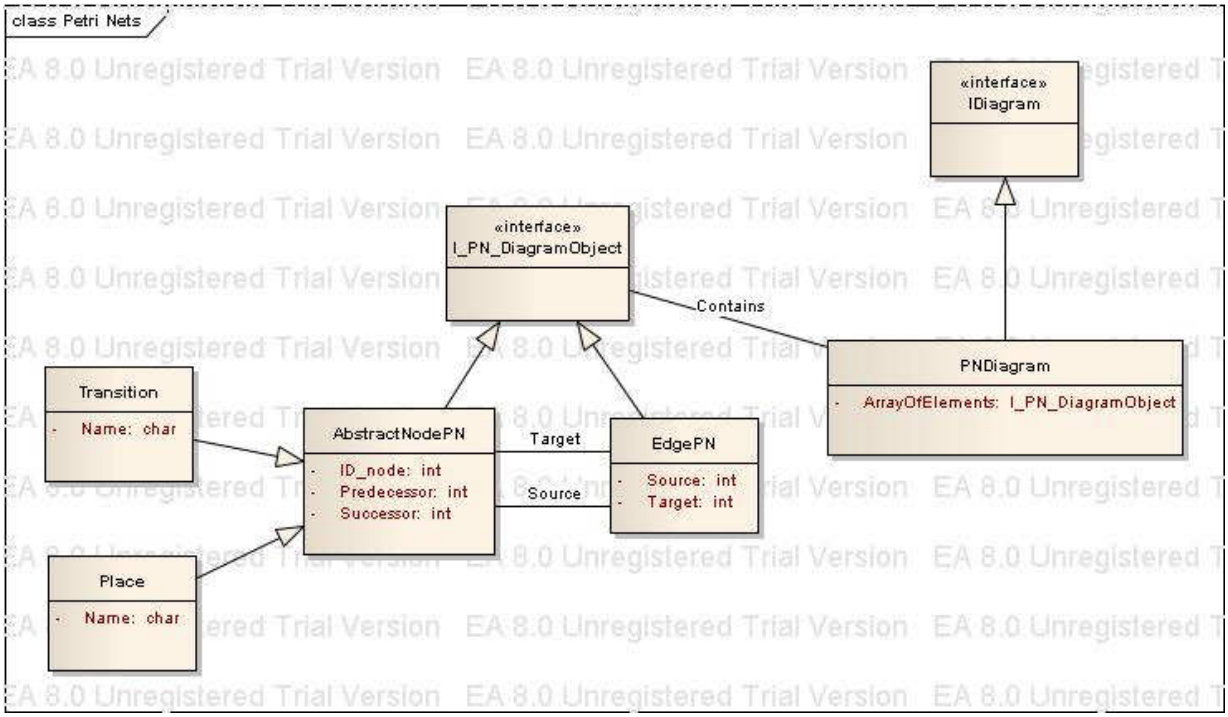


Obrázek 35 - Upravený EPC Metamodel

3.2.5 Metamodel Petriho sítě

Pomocí vytvořeného metamodelu Petriho sítí budeme schopni vytvářet modely, které budou obsahovat následující elementy

- Place (Místo)
- Transition (Přechod)
- Sekvenční (kontrolní) tok (Edge, Arc, harana), který spojuje jednotlivé elementy

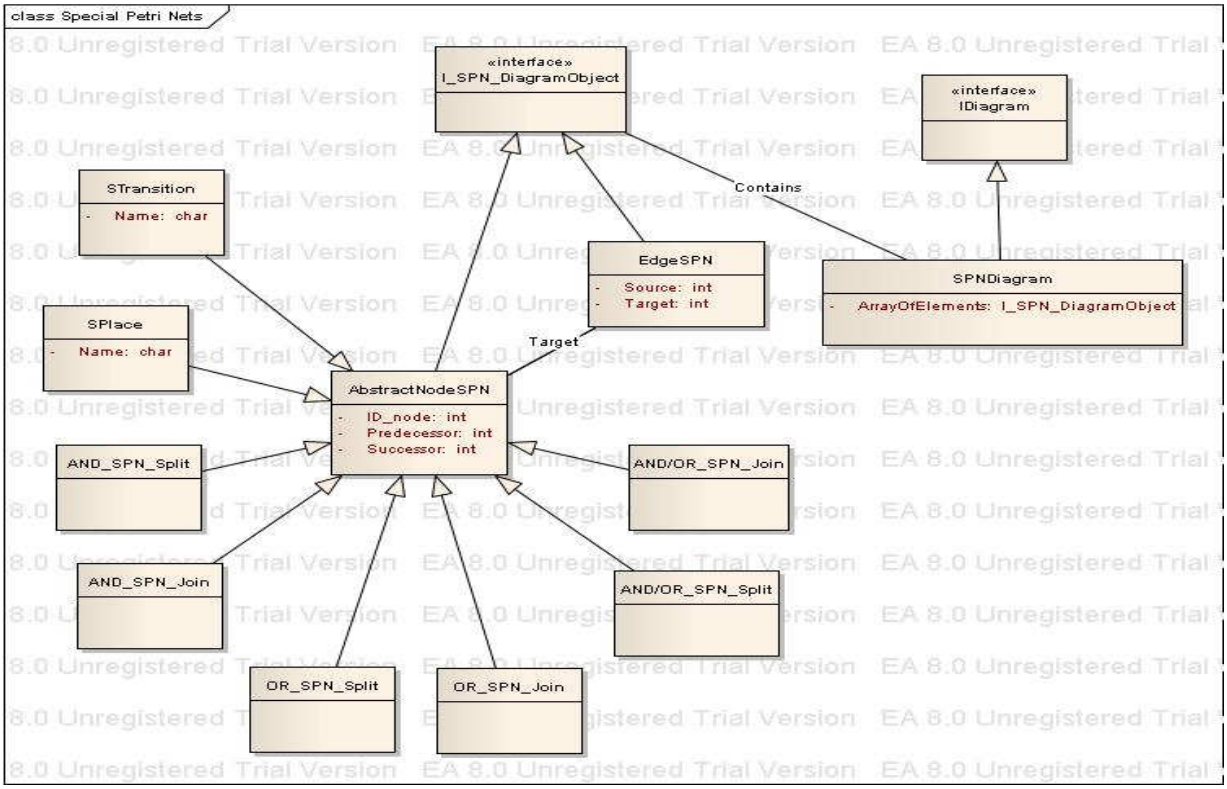


Obrázek 36 - Upravený Petriho sítě Metamodel

3.2.6 Metamodel speciál Petriho sítě

Pomocí vytvořeného metamodelu speciál Petriho sítí budeme schopni vytvářet modely, které budou obsahovat následující elementy. Na rozdíl od předešlé Petriho sítě, tento metamodel navíc definuje elementy pro logické spojky. Zdůvodnění, proč jsme tuto notaci zavedli, a převodní pravidla jsou uvedena v kapitole převodních pravidel Petriho sítí.

- Place (Místo)
- Transition (Přechod)
- Sekvenční (kontrolní) tok (Edge, Arc, hrana), který spojuje jednotlivé elementy
- Logická spojka AND typu split
- Logická spojka AND typu join
- Logická spojka OR typu split
- Logická spojka OR typu join
- Logická spojka AND/OR typu split
- Logická spojka AND/OR typu join

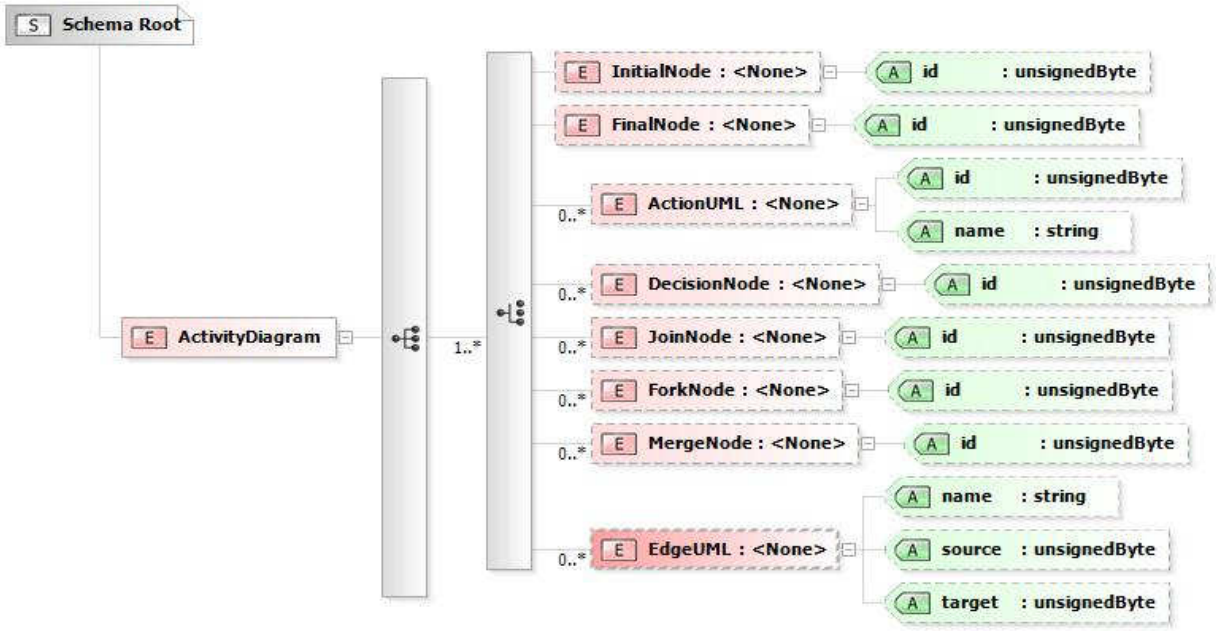


Obrázek 37 - Upravený speciál Petriho sítě Metamodel

3.3 XML Schéma

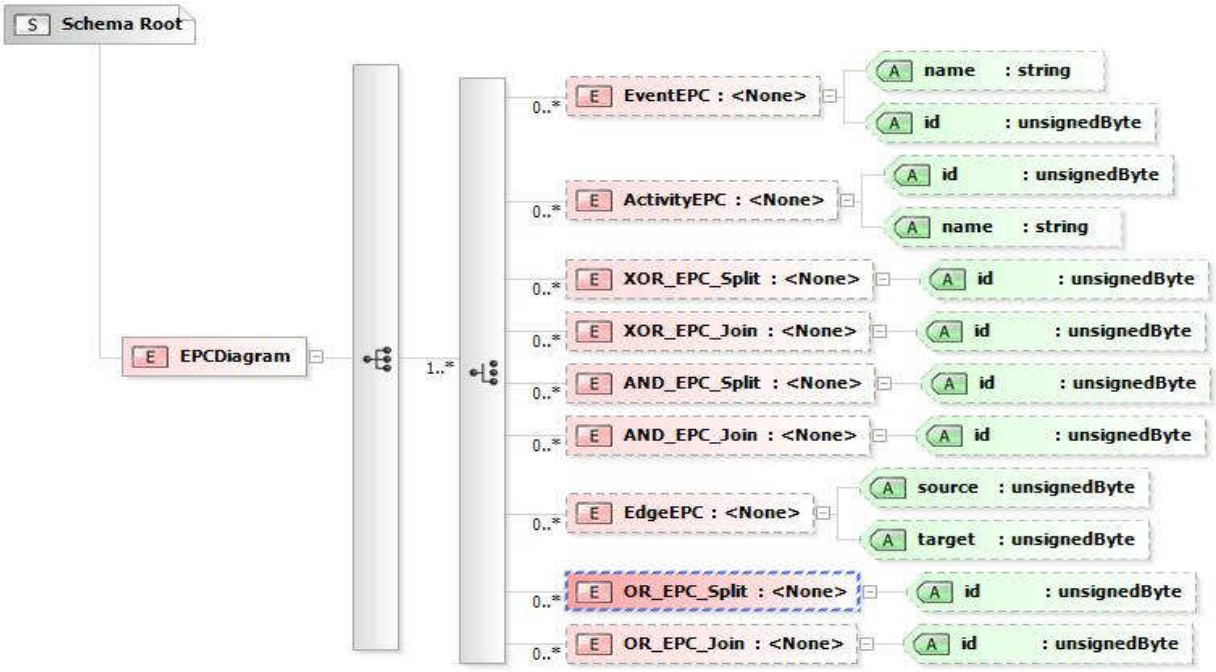
Kompletní zdrojové kódy všech XML Schemat naleznete v příloze. Pro lepší orientaci, je zde vyobrazena pouze grafická podoba pro každou ze čtyř notací, které používáme na vstupu transforace.

3.3.1 Aktivitní diagram (UML) - XML Schema



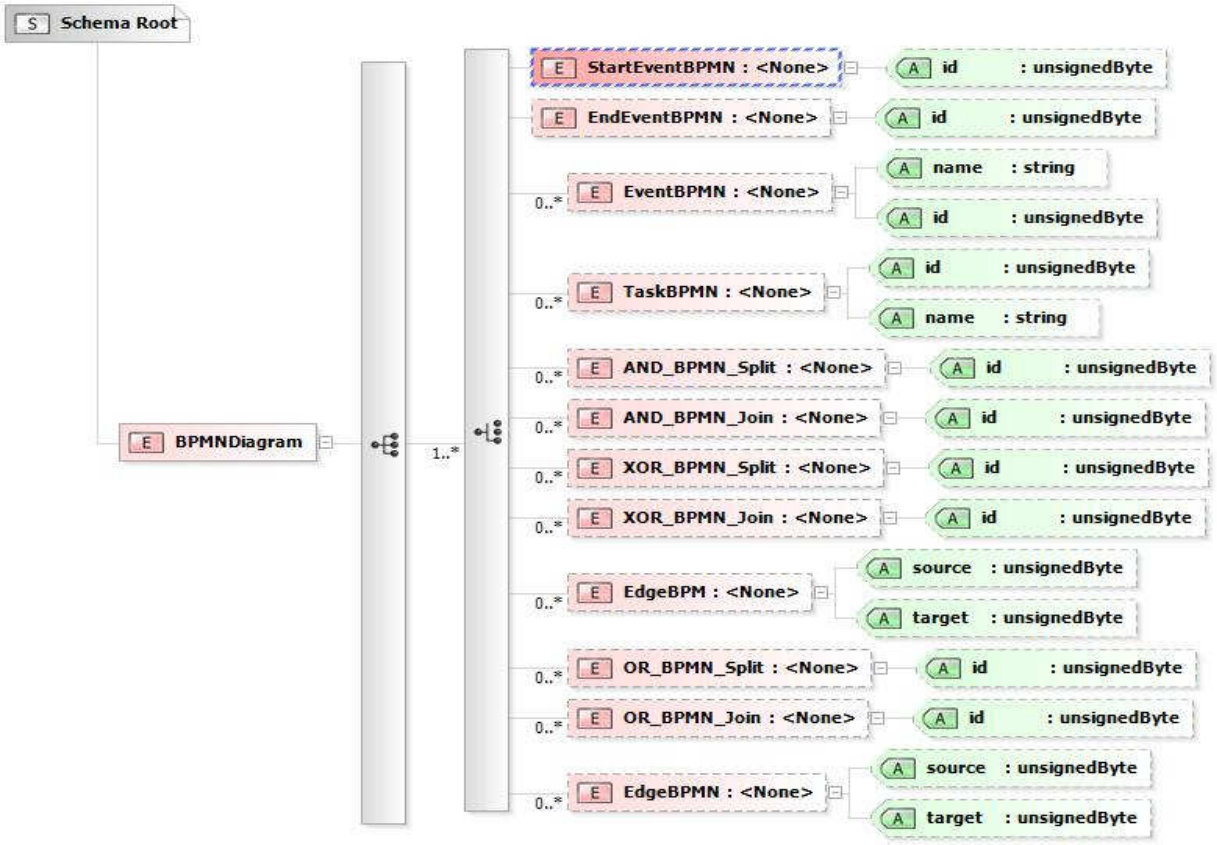
Obrázek 38- Aktivitní diagram (UML) XML Schema

3.3.2 EPC – XML Schema



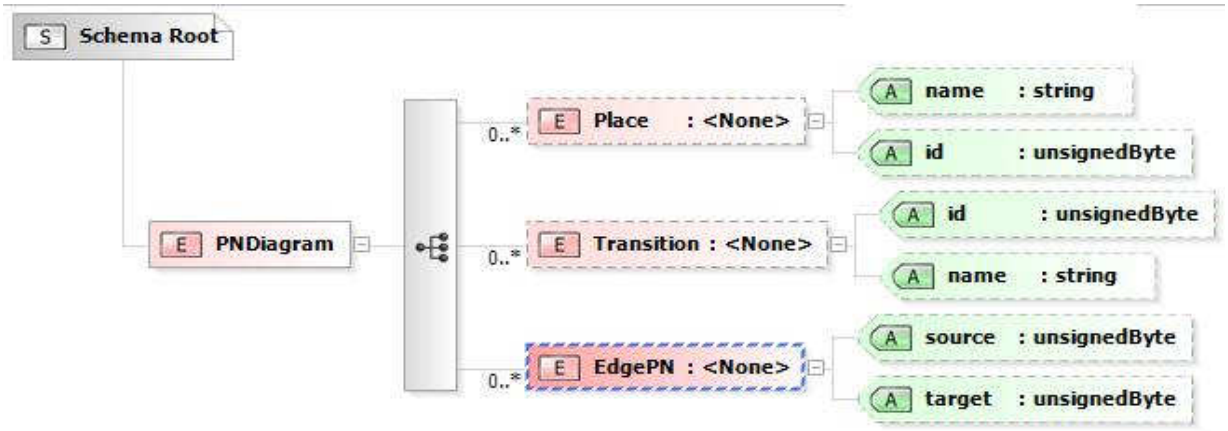
Obrázek 39 - EPC XML Schema

3.3.3 BPMN – XML Schema



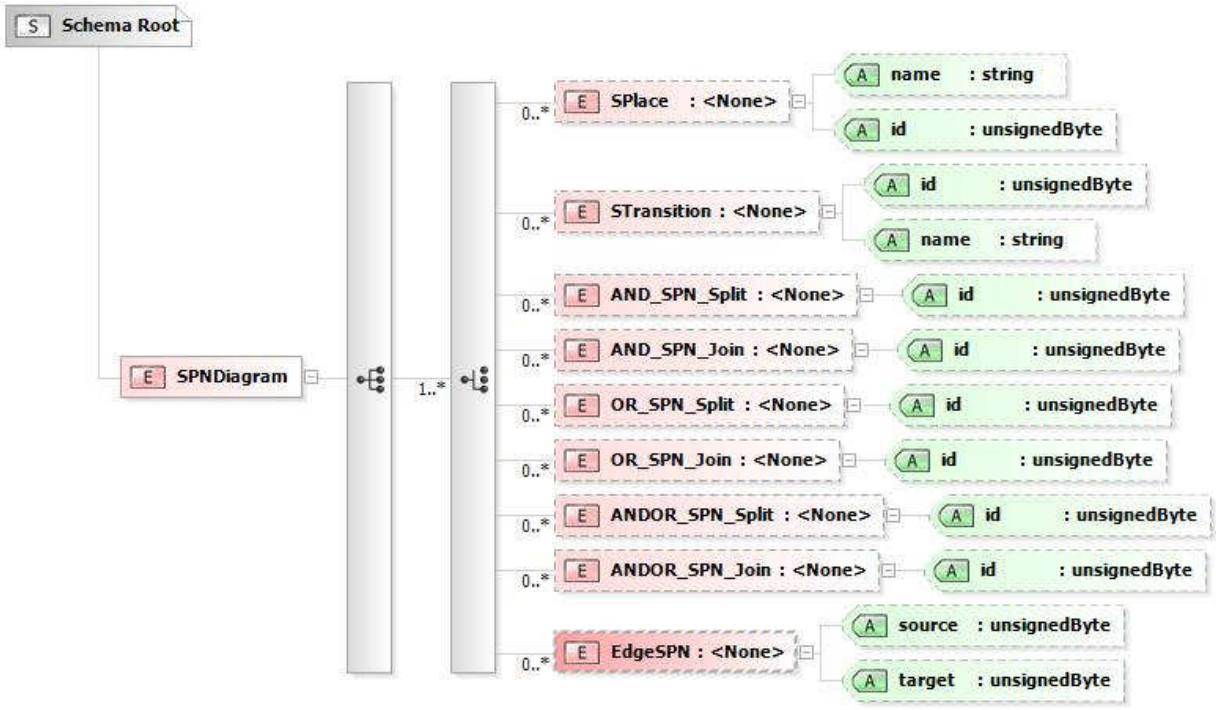
Obrázek 40 - BPMN XML Schema

3.3.4 Petriho sítě – XML Schema



Obrázek 41 - Petriho sítě XML Schéma

3.3.5 Speciální Petriho sítě – XML Schema



Obrázek 42 - Speciál Petriho sítě XML Schéma

3.4 Převodní pravidla

Jak již bylo řečeno, jednotlivé transformace se budou realizovat vždy mezi zvolenou notací a námi vytvořeným mezi-metamodelem. Tzn., pokud například budeme chtít provést převod mezi Aktivitním diagramem UML a Petriho sítěmi, bude transformace vypadat následovně:

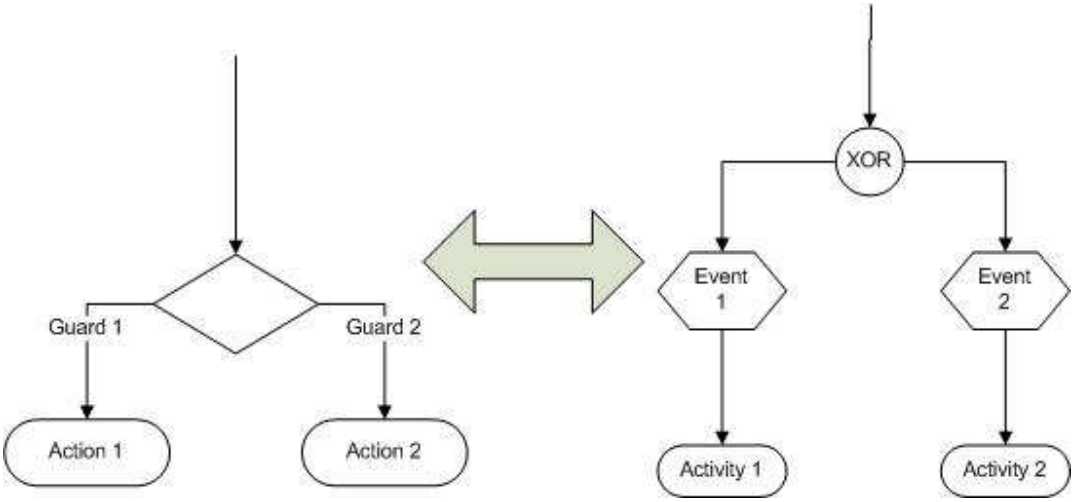
1. Převedeme UML model na model mezi-modelu
2. Převedeme model mezi-modelu na Petriho sítě

Tudíž následující převodní pravidla budou definována vždy mezi zvolenou notací a notací, která vychází z mezi-metamodelu. Převodní pravidla, která jsou zobrazena v níže uvedených tabulkách, jsou použita pro transformace v obou směrech, až na výjimku u Petriho sítí, kde jsme zavedli dvojí notaci, jednu pro převod mezi mezi-metamodel a Petriho sítěmi a druhou mezi speciálně zavedenou Petriho sítí a mezi-metamodelem.

3.4.1 Převodní pravidla UML - Aktivitní diagram

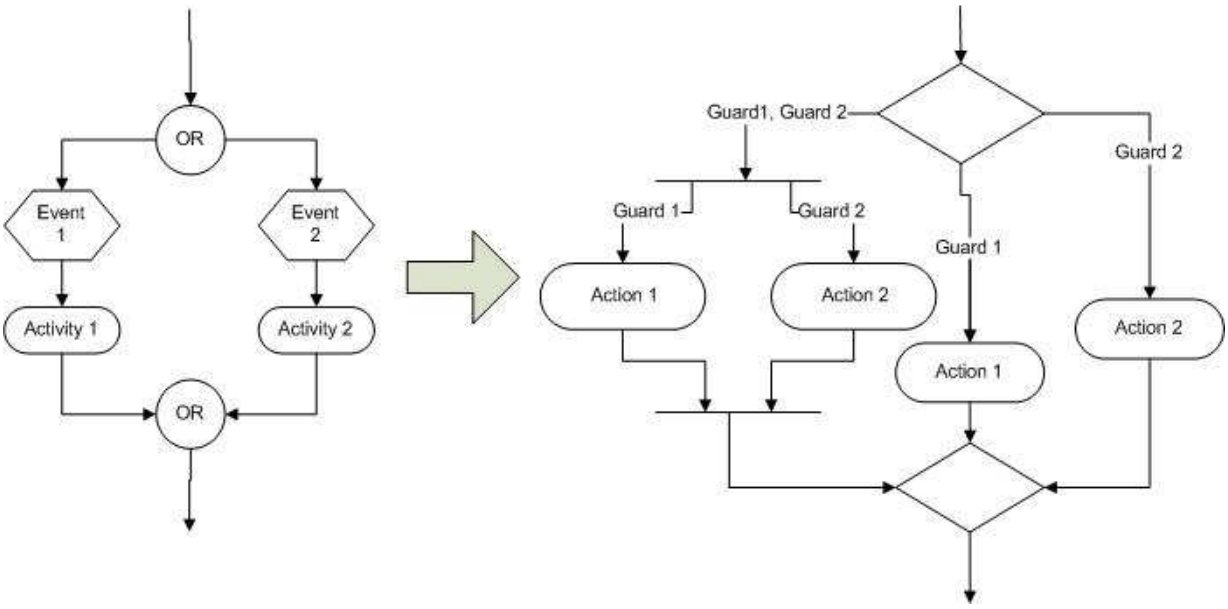
U aktivitního diagramu je třeba zmínit, že umožňuje slučování více vstupů u aktivit na rozdíl od ostatních notací jako je např. EPC. Tudíž není nutné použít element XOR Join, neboli Join Node ke sloučení alternativních toků. Tato vlastnost se může stát výhodou úspornějšího grafického znárodnění, ale pokud chceme dodržovat kvalitní strukturu při modelování Aktivitního diagramu, je doporučeno XOR Join používat, což bude i náš případ. Transformace je realizována dle převodní tabulky v obou směrech.

Převod strážních podmínek u UML se přetransformuje na událost mezi-metamodelu, což obnáší jisté korekce kontrolní toků. Transformace strážní podmínky Aktivitního diagramu na mezi-metamodel je popsána na obrázku 43. Při opačné transformaci se bude postupovat zpětných způsobem.

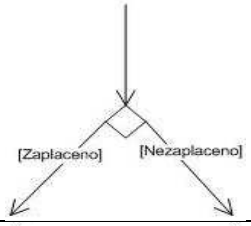
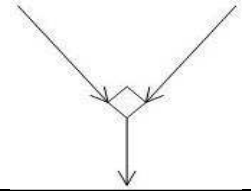


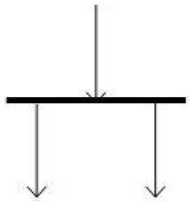
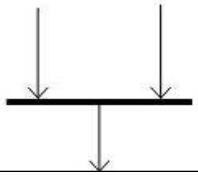
Obrázek 43 - Převod strážních podmínek Aktivitního diagramu (UML)

Faktem je, že Aktivitní diagram jako jediná z vybraných notací neobsahuje možnost modelování logické spojky OR. Je sice pravdou, že tato spojka je ze všech logických operátorů využívána nejméně a ve zdrojích popisující transformace je většinou vynechávána, přesto se pokusíme vytvořit alternativu pomocí ostatních dvou logických spojek a to větvení (OR) a synchronizace (AND) . Převod z transformačního mezi-metamodelu na Aktivitní diagram, tak jak jej bude realizovat aplikace, nám demonstruje obrázek 44. Z důvodu, že tento převod není jednoduchý, omezíme možnost transformace pouze na dvě výstupní větve vedoucí z logické spojky.



Obrázek 44 - Převod logické spojky OR v aktivitním diagramu (UML)

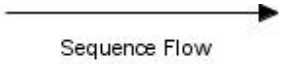
UML – Aktivitní diagram	Notace mezi-metamodelu
	**Startovací element**
	**Ukončovací element**
	**Aktivita**
[strážní podmínka sekvenčního toku]	**Událost**
	Sekvenční tok
	**Rozhodovací blok (XOR)- split** **(včetně strážních podmínek)**
	**Rozhodovací blok (XOR) – join**

	**Paralelní tok (AND) – split**
	**Paralelní tok (AND)- join**

Tabulka 2 - Převodní pravidla mezi Aktivitním diagramem (UML), mezi-metamodelem a zpět

3.4.2 Převodní pravidla BPMN - Business Process Diagram (BPD)

BPMN díky své bohaté sadě elementů, je téměř totožné s naším transformačním mezi-metamodelem, tudíž zde převedeme jednotlivé elementy dle tabulky. Pouze při převodu z BPMN do transformačního metamodelu, budeme muset u logických spojek testovat dle počtu vstupů a výstupů, jestli se jedná o typ split nebo join a dle výsledku testu použít patřičné převodní pravidlo.

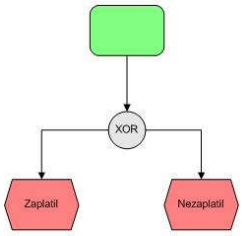
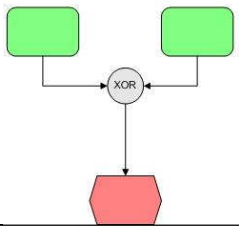
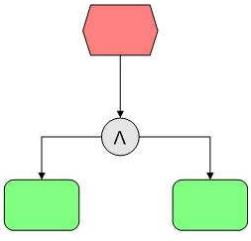
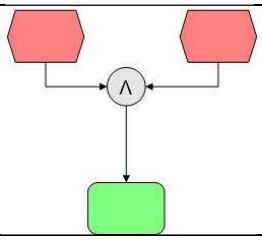
BPMN – Business Process Diagram (BPD)	Notace mezi-metamodelu
	**Startovací element**
	**Ukončovací element**
	**Aktivita**
	**Událost**
	**Sekvenční tok**
	**Rozhodovací blok (XOR) - split** **(včetně strážních podmínek)**

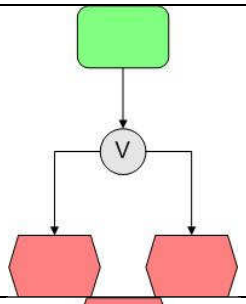
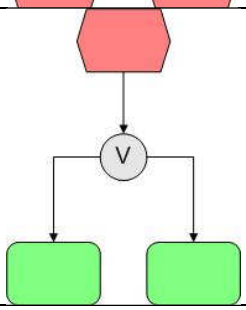
	**Rozhodovací blok (XOR)- join**
	**Paralelní tok (AND) – split**
	**Paralelní tok (AND)- join**
	**Rozhodovací blok (OR) – split**
	**Rozhodovací blok (OR) – join**

Tabulka 3 - Převodní pravidla mezi BPMN, mezi-metamodelem a zpět

3.4.3 Převodní pravidla EPC

Při transformaci na EPC je třeba myslet na to, aby byly dodrženy konvence EPC diagram. Prvním pravidlem, které říká, že EPC digram musí začínat i končit událostí si poradíme zcela jednoduše tím, startovací a ukončovací element transformačního mezi-metamodelu převedeme na událost v EPC. Dále je v EPC nutné dodržovat střídání událostí a aktivit, tuto podmínku vyřešíme pomocí testu, který bude naimplementován v aplikaci. Tento test projede celý diagram element po element a bude testovat, zda byl diagram převeden korektně, pokud test zjistí, že se vedle sebe vyskytují dvě události nebo dvě aktivity, vytvoří speciální fake element (značka nebude mít žádnou hodnotu) opačného typu a vloží jej mezi dva stejné elementy. Další podmínkou je, že událost je pasivní element, který nemůže být vstupem do rozhodovacího bloku. To dělá vždy jen aktivita. Pro dodržení této podmínky bude naimplementován další test, který v případě výskytu události před logickou spojkou XOR split nebo OR split doplní již zmiňovanou fake aktivitu. ID v XML dokumentu se vždy shodují s ID v diagramu.

EPC	Notace mezi-metamodelu
	**Startovací element**
	**Ukončovací element**
	**Aktivita**
	**Událost**
	**Sekvenční tok**
	**Rozhodovací blok (XOR) - split** **(včetně strážních podmínek)**
	**Rozhodovací blok (XOR) - join**
	**Paralelní tok (AND) - split**
	**Paralelní tok (AND) - join**

	**Rozhodovací blok (OR) - split** **(včetně strážních podmínek)**
	**Rozhodovací blok (OR) - join**

Tabulka 4 - Převodní pravidla mezi eEPC, mezi-metamodelem a zpět

3.4.4 Převodní pravidla Petriho sítě

U Petriho sítí se při převodu budeme muset zaměřit na několik věcí. První záležitostí je, že si zavedeme dvojí pravidla, jedny pro transformaci z transformačního metamodelu na Petriho sítě a druhé pro opačný proces. A to z důvodu problematického převodu logických spojek v kombinaci s pravidly pro tvorbu diagramů pomocí Petriho sítí. Navíc si pro převod z Petriho sítí na transformační model zavedeme speciální notaci (Speciál Petriho síť), která bude stále vycházet z definice Petriho sítí a navíc se svou strukturou bude podobat ostatním notacím (viz tabulka 5). V Petriho sítích nejsou zavedeny konkrétní elementy pro logické spojky, tudíž si musíme vystačit s kombinací míst, přechodů a hran, které nám jednotlivé logické spojky definují. Pro zjednodušení realizace vstupního modelu Petriho sítí si v nově vytvořené notaci logické spojky zavedeme dle následujících pravidel a dle transformačních pravidel uvedených v tabulce budeme schopni převést.

Zajímavým poznatkem u Petriho sítí je význam logických spojek OR a AND/Split. U Petriho sítí je význam těchto spojek odlišný, než jak jsme byli zvyklí u ostatních notací. Tzn., že OR zde používáme pro alternativní větvení a AND/OR pro výběr jedné, druhé či obou větví.

V případě opačné transformace budeme jednotlivé logické spojky převádět dle pravidel uvedených v tabulce pro převod z transformačního modelu na Petriho sítě.

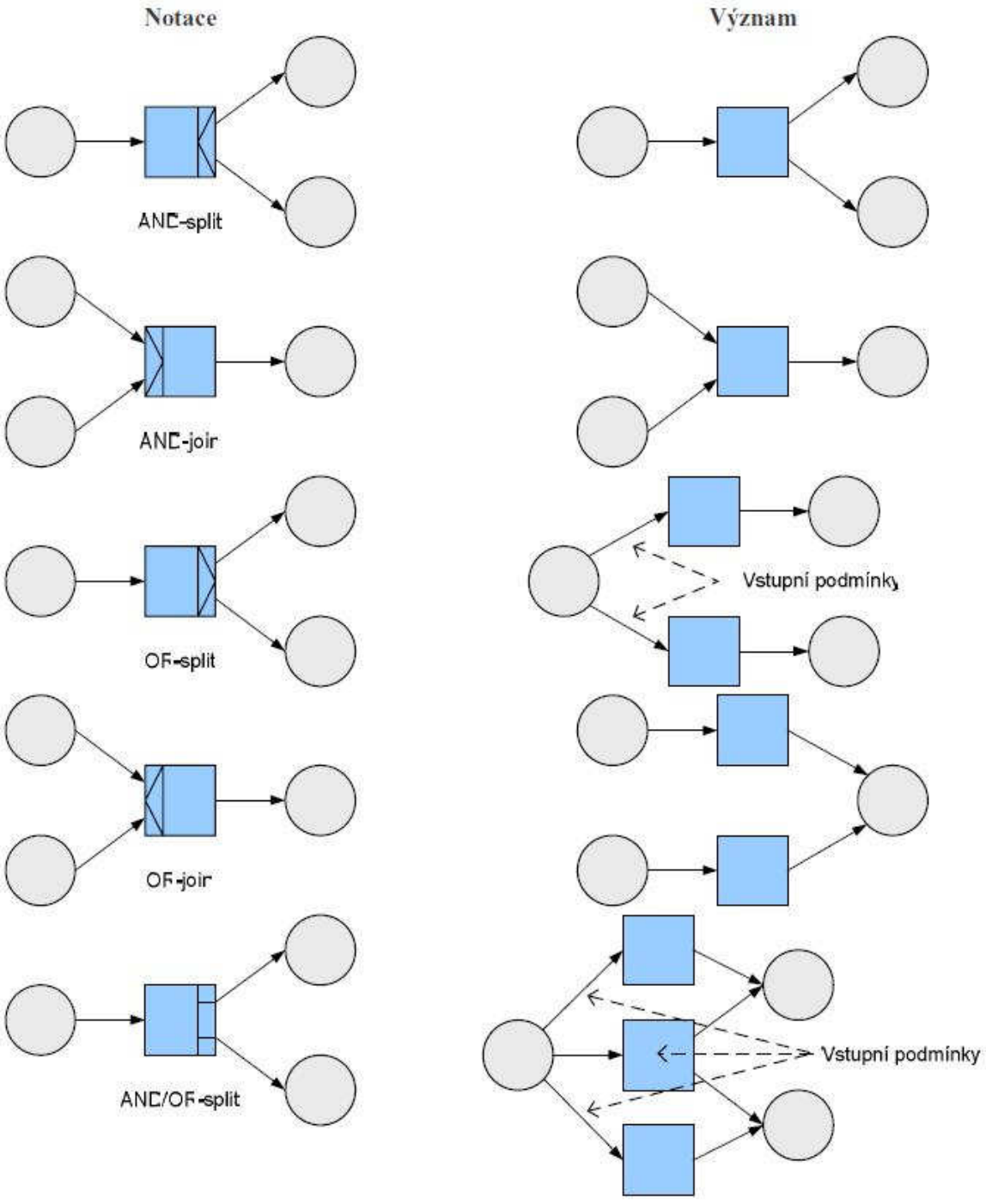
Další důležitou informací je fakt, že pomocí základních elementů Petriho sítí se těžko modelují procesy, které jsme schopni namodelovat v ostatních notacích, z tohoto důvodu budeme transformace realizovat pomocí barevného rozlišení, které se označuje jako Barevné Petriho sítě.

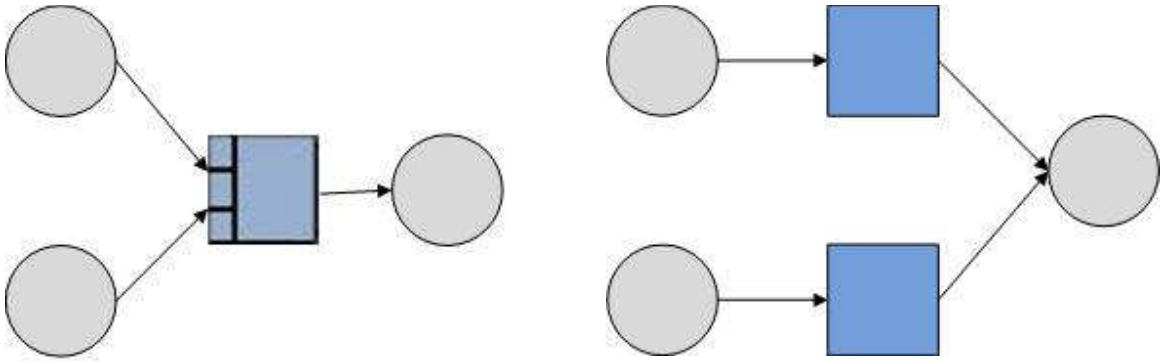
V případě rozšíření Petriho sítí o barvy se jedná o přiřazení hodnoty (barvy) značce. Díky tomu lze mezi jednotlivými značkami rozlišovat co vyjadřují a provádění přechodů může probíhat na základě informace, která není explicitně vyjádřena v grafu Petriho sítě, ale je ukryta v hodnotách značek a v procedurách spojených s prováděním přechodů. To znamená, že na rozdíl od klasických Petriho sítí proveditelný přechod nemusí být proveden, poněvadž hodnota značky nesplňuje podmínky jeho provedení. Přítomnost značek na vstupních místech tedy nepostačuje k provedení přechodu. Obdobně přechod nemusí produkovat značky na všechna svá výstupní místa, ale pouze na ta, která jsou určena hodnotou spotřebované značky a předdefinovanými pravidly provedení přechodu.

(Převzato z http://vondrak.cs.vsb.cz/download/Metody_byznys_modelovani.pdf)

Stejně jako u notace EPC je třeba dodržet pravidlo, že každý diagram je započat i zakončen událostí. V Petriho sítích budeme mapovat události na element Place a aktivity na element Transition.

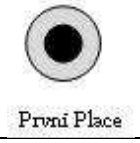
Další podmínkou je, že v digramu musí být striktně dodrženo pravidlo střídání událostí s aktivitami. Pro splnění této podmínky využijeme podobné operace jako u notace EPC, kdy výsledný model při převodu z transformačního modelu na Petriho sítě podrobíme testu a v případě, že by tato podmínka nebyla splněna, budeme muset doplnit fake elementy.


Notace
Význam
AND-split
AND-join
OR-split
OR-join
AND/OR-split
Vstupní podmínky
Vstupní podmínky

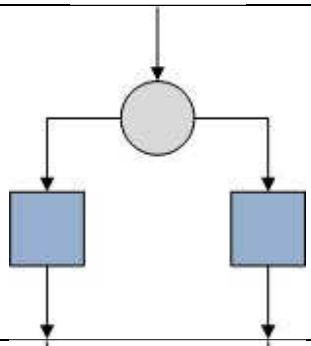
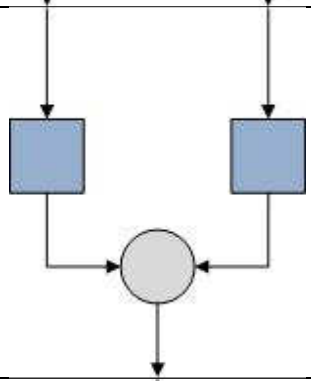
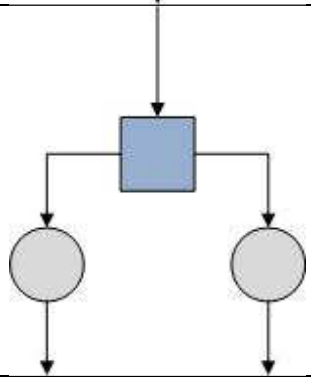


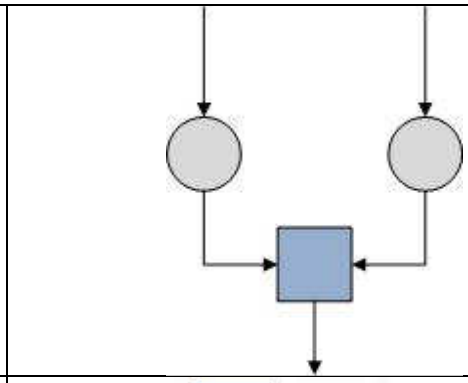
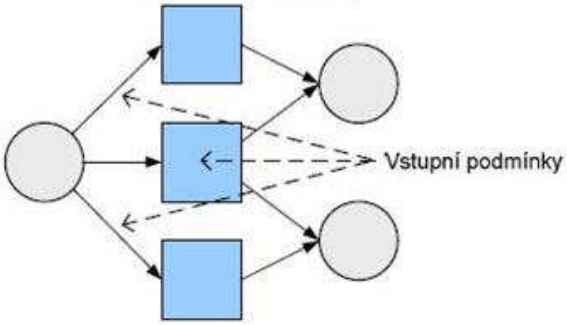
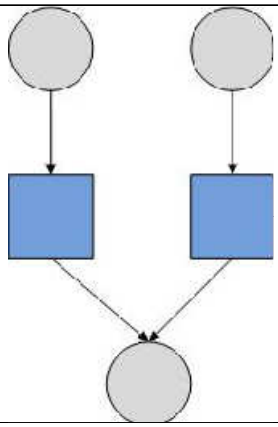
Tabulka 5 - Převodní pravidla pro zavedení speciální Petriho sítě notace

(Převzato z http://vondrak.cs.vsb.cz/download/Metody_byznys_modelovani.pdf)

Special Petriho Sítě	Notace mezi-metamodelu
	**Startovací element**
	**Ukončovací element**
	**Aktivita**
	**Událost**
	**Sekvenční tok**
	**Rozhodovací blok (XOR) - split** **(včetně strážních podmínek)**
	**Rozhodovací blok (XOR) – join**
	**Paralelní tok (AND) – split**
	**Paralelní tok (AND) – join**
	**Rozhodovací blok (OR) – split**
	**Rozhodovací blok (OR) – join**

Tabulka 6 - Převodní pravidla pro transformaci ze speciální Petriho sítí notace na mezi- metamodel

Notace mezi-metamodelu	Barevná petriho síť
Startovací element	
Ukončovací element	
Aktivita	
Událost	
Datový tok	
Rozhodovací blok (XOR) - split **(včetně strážních podmínek)**	
Rozhodovací blok (XOR) - join	
Paralelní tok (AND) - split	

Paralelní tok (AND) - join	
Rozhodovací blok (OR) - split	
Rozhodovací blok (OR) - join	

Tabulka 7 - Převodní pravidla pro transformaci z mezi-metamodelu do Petriho sítí

3.5 Aplikace pro transformaci mezi notací

Jak už bylo zmíněno v zadání, jedná se o aplikaci, která realizuje transformace mezi notacemi UML (Aktivitní diagram), BPMN, EPC a Petriho sítěmi prostřednictvím transformačního modelu. Aplikace je naimplementována v programovacím jazyce Java a prostředí Eclipse.

Obsluha této aplikace je velmi jednoduchá a skládá se z následujících několika kroků. Pro spuštění aplikace je třeba zadat 4 argumenty:

- Vstupní notace – zde je nutné zadat jednu z vybraných notací, ze které chceme transformaci uskutečnit. Vstupní formát musím být zadán v následujících formátech (UML, BPMN, EPC, Petriho Sítě)
- Cesta k XML dokumentu modelu, který budeme transformovat - Pro uskutečnění transformace je třeba mít vyhotoven XML dokument, který vychází z XML schema notace, a zároveň, který chceme zadat na vstupu, a navíc popisuje námi vytvořený model.
- Výstupní notace – zde zadáme notaci, na kterou chceme transformovat. Výstupní formát musí být zadán následovně (UML, BPMN, EPC, Petriho Sítě)
- Cesta výstupu - kde se má vygenerovat výstupní XML dokument

Pokud je jeden nebo více argumentů zadán chybně, nedojde ke spuštění programu a budete opakovaně vyzváni k zadání argumentů. Pokud jsou všechny argumenty zadány korektně, aplikace se spustí a provede následující kroky:

- načte vstupní XML dokument
- provede validaci, jestli obsah načteného dokumentu odpovídá struktuře, která definuje zadanou notaci jako Vstupní notace
- pokud ne, budete o této chybě informování na výstupu a budete opakovaně vyzváni k zadání argumentů
- pokud ano, aplikace provede transformaci na transformační model a poté provede transformaci na zvolenou výstupní notaci
- výsledný XML kód se zobrazí na výstupu a zároveň se vygeneruje XML dokument se shodným obsahem, jako je zobrazen na výstupu. Výsledný XML dokument je vytvořen na místě dle zadané výstupní cesty.

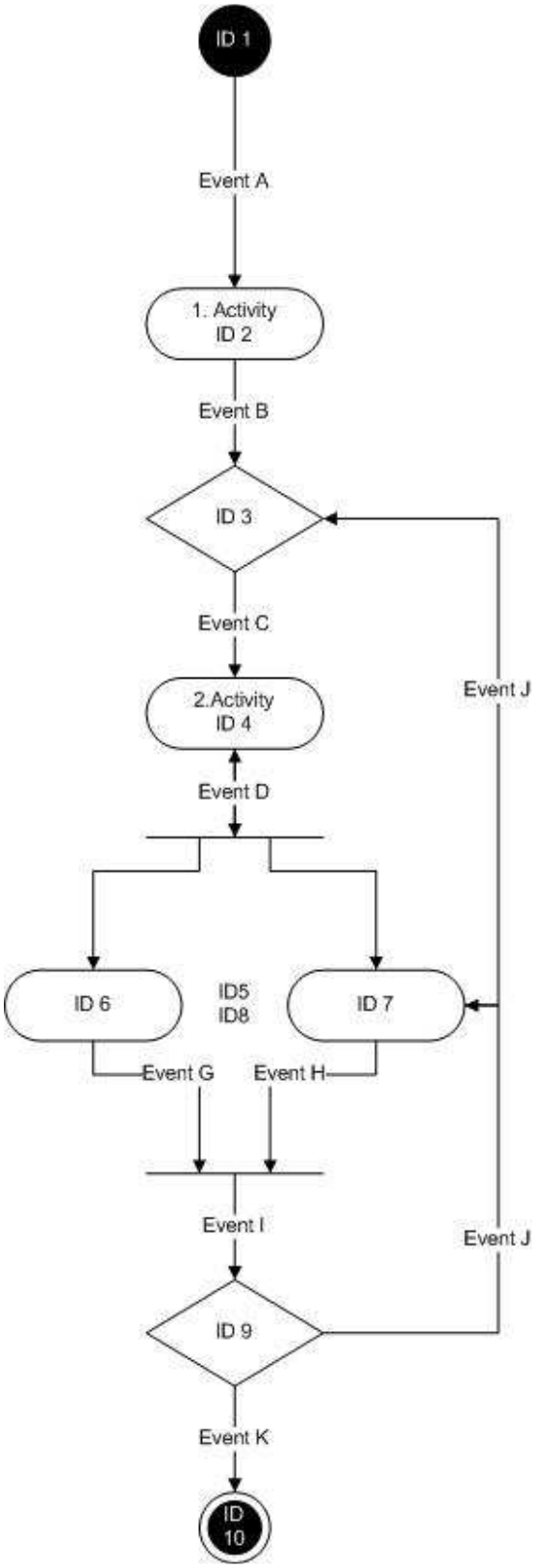
Detailnější informace o struktuře aplikace najdete v programátorské příručce, která je součástí příloh této práce.

3.6 Vzorové vstupní modely

Pro lepší představu, jak by měl vstupní model vypadat, jsme namodelovali pro každou notaci příklad vstupního modelu. Poněvadž do aplikace budeme načítat model ve formátu XML dokument, který není zcela jednoduše čitelný na první pohled, je každý model zobrazen jak v XML dokumentu tak i v grafické podobě příslušného diagramu. V diagramech jsou navíc přiřazena ID, která identifikují při zpracování aplikací dané elementy. V případě, že v transformaci dojde k přidání elementů, které nebyly načteny ze vstupu, například u Petriho sítí či EPC elementy typu Fake, z důvodu zachování pravidel, která definují danou notaci, jsou jejich ID´s generována vždy od první nejmenší volné hodnoty. Pro vytvoření XML dokumentů vycházející z výše definovaných XML schemat byl využit software Oxygen XML Editor. Většina těchto vstupních modelů byla navržena tak, aby pokryla kompletní funkčnost dané notací, která vychází z námi vytvořených metamodelů.

3.6.1 Aktivitní diagram (UML)

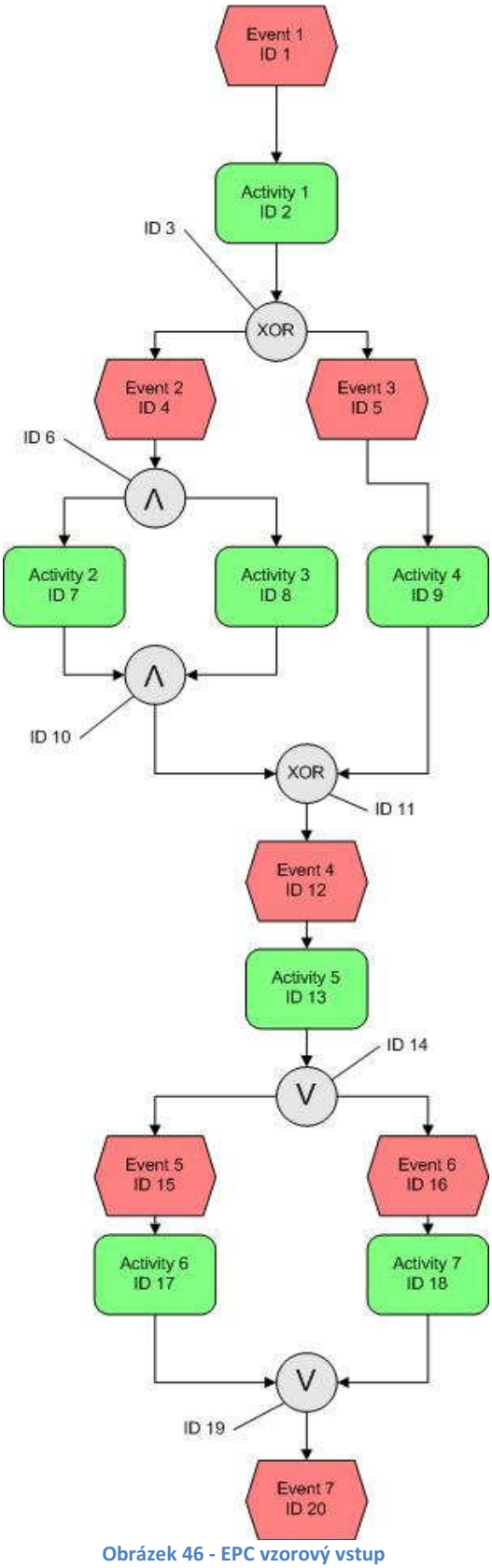
```
<?xml version="1.0" encoding="utf-8"?>
<ActivityDiagram>
  <InitialNode id="1"/>
  <FinalNode id="10"/>

  <ActionUML id="2" name="Activity 1"/>
  <ActionUML id="4" name="Activity 2"/>
  <ActionUML id="6" name="Activity 3"/>
  <ActionUML id="7" name="Activity 4"/>

  <DecisionNode id="9"/>
  <JoinNode id="3"/>

  <ForkNode id="5"/>
  <MergeNode id="8"/>

  <EdgeUML name="edgeA" source="1" target="2"/>
  <EdgeUML name="edgeB" source="2" target="3"/>
  <EdgeUML name="edgeC" source="3" target="4"/>
  <EdgeUML name="edgeD" source="4" target="5"/>
  <EdgeUML name="" source="5" target="6"/>
  <EdgeUML name="" source="5" target="7"/>
  <EdgeUML name="G" source="6" target="8"/>
  <EdgeUML name="H" source="7" target="8"/>
  <EdgeUML name="I" source="8" target="9"/>
  <EdgeUML name="edgeJ" source="9" target="3"/>
  <EdgeUML name="edgeK" source="9" target="10"/>
</ActivityDiagram>
```



Obrázek 45 - Aktivitní diagram (UML) vzorový vstup

3.6.2 EPC

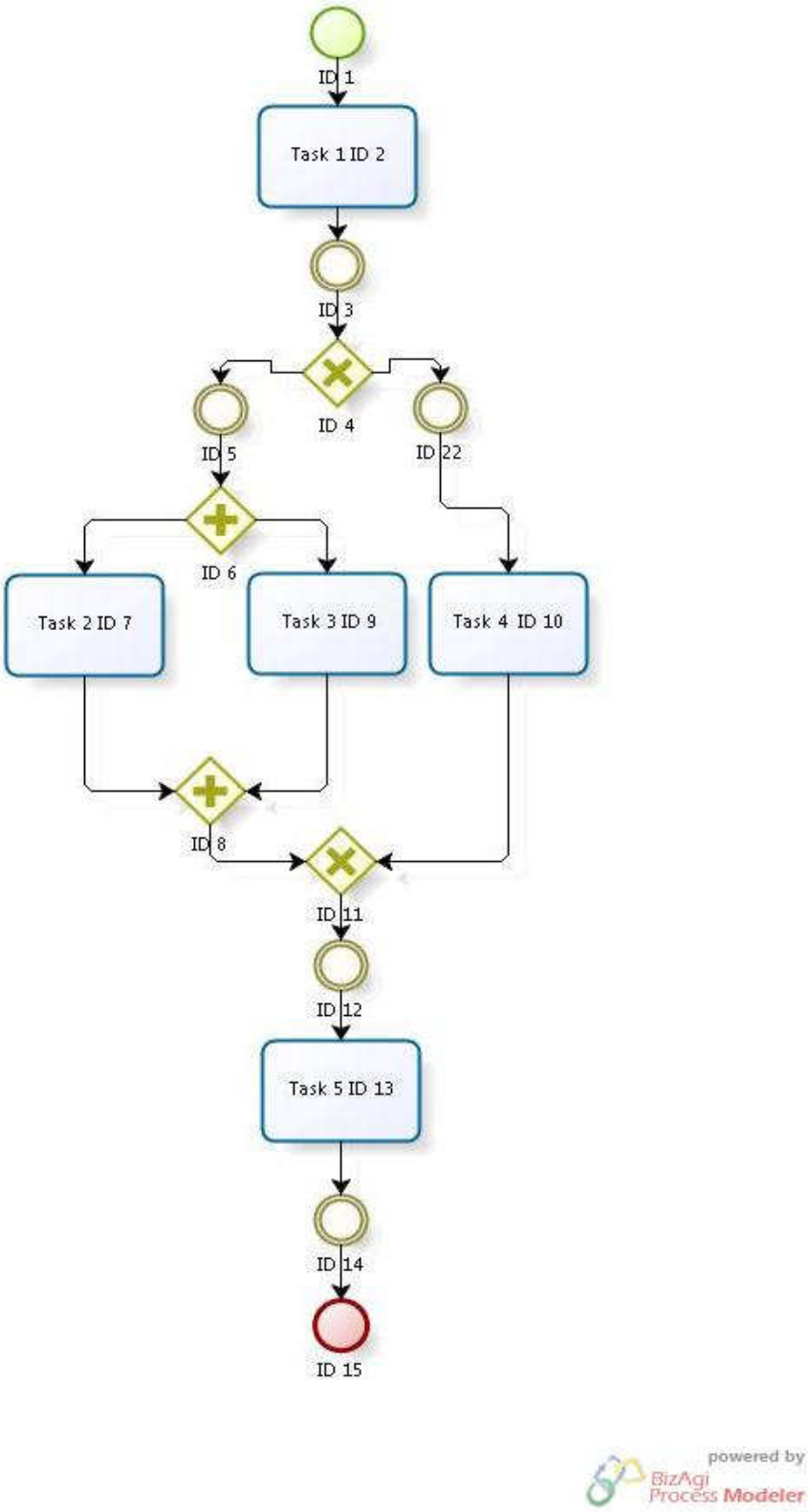
```
<?xml version="1.0" encoding="utf-8"?>
<EPCDiagram>
  <EventEPC name="Event1" id="1"/>
  <EventEPC name="Event2" id="4"/>
  <EventEPC name="Event3" id="5"/>
  <EventEPC name="Event4" id="12"/>
  <EventEPC name="Event5" id="15"/>
  <EventEPC name="Event6" id="16"/>
  <EventEPC name="Event7" id="20"/>

  <ActivityEPC id="2" name="Activity 1"/>
  <ActivityEPC id="7" name="Activity 2"/>
  <ActivityEPC id="8" name="Activity 3"/>
  <ActivityEPC id="9" name="Activity 4"/>
  <ActivityEPC id="13" name="Activity 5"/>
  <ActivityEPC id="17" name="Activity 6"/>
  <ActivityEPC id="18" name="Activity 7"/>

  <XOR_EPC_Split id="3"/>
  <XOR_EPC_Join id="11"/>

  <AND_EPC_Split id="6"/>
  <AND_EPC_Join id="10"/>

  <OR_EPC_Split id="14"/>
  <OR_EPC_Join id="19"/>

  <EdgeEPC source="1" target="2"/>
  <EdgeEPC source="2" target="3"/>
  <EdgeEPC source="3" target="4"/>
  <EdgeEPC source="4" target="6"/>
  <EdgeEPC source="6" target="7"/>
  <EdgeEPC source="7" target="10"/>
  <EdgeEPC source="6" target="8"/>
  <EdgeEPC source="8" target="10"/>
  <EdgeEPC source="10" target="11"/>
  <EdgeEPC source="3" target="5"/>
  <EdgeEPC source="5" target="9"/>
  <EdgeEPC source="9" target="11"/>
  <EdgeEPC source="11" target="12"/>
  <EdgeEPC source="12" target="13"/>
  <EdgeEPC source="13" target="14"/>
  <EdgeEPC source="14" target="15"/>
  <EdgeEPC source="15" target="17"/>
  <EdgeEPC source="17" target="19"/>
  <EdgeEPC source="14" target="16"/>
  <EdgeEPC source="16" target="18"/>
  <EdgeEPC source="18" target="19"/>
  <EdgeEPC source="19" target="20"/>


</EPCDiagram>
```



Obrázek 46 - EPC vzorový vstup

3.6.3 BPMN

```
<?xml version="1.0" encoding="utf-8"?>

<BPMNDiagram>
  <StartEventBPMN id="1"/>
  <EndEventBPMN id="15"/>

  <EventBPMN name="Event 1" id="3"/>
  <EventBPMN name="Event 2" id="5"/>
  <EventBPMN name="Event 3" id="22"/>
  <EventBPMN name="Event 4" id="12"/>
  <EventBPMN name="Event 5" id="14"/>

  <TaskBPMN id="2" name="Task 1"/>
  <TaskBPMN id="7" name="Task 2"/>
  <TaskBPMN id="9" name="Task 3"/>
  <TaskBPMN id="10" name="Task 4"/>
  <TaskBPMN id="13" name="Task 5"/>

  <AND_BPMN_Split id="6"/>
  <AND_BPMN_Join id="8"/>

  <XOR_BPMN_Split id="4"/>
  <XOR_BPMN_Join id="11"/>

  <EdgeBPM source="1" target="2"/>
  <EdgeBPM source="2" target="3"/>
  <EdgeBPM source="3" target="4"/>
  <EdgeBPM source="4" target="5"/>
  <EdgeBPM source="5" target="6"/>
  <EdgeBPM source="6" target="7"/>
  <EdgeBPM source="7" target="8"/>
  <EdgeBPM source="6" target="9"/>
  <EdgeBPM source="9" target="8"/>
  <EdgeBPM source="8" target="11"/>
  <EdgeBPM source="4" target="22"/>
  <EdgeBPM source="22" target="10"/>
  <EdgeBPM source="10" target="11"/>
  <EdgeBPM source="11" target="12"/>
  <EdgeBPM source="12" target="13"/>
  <EdgeBPM source="13" target="14"/>
  <EdgeBPM source="14" target="25"/>


 </BPMNDiagram>
```

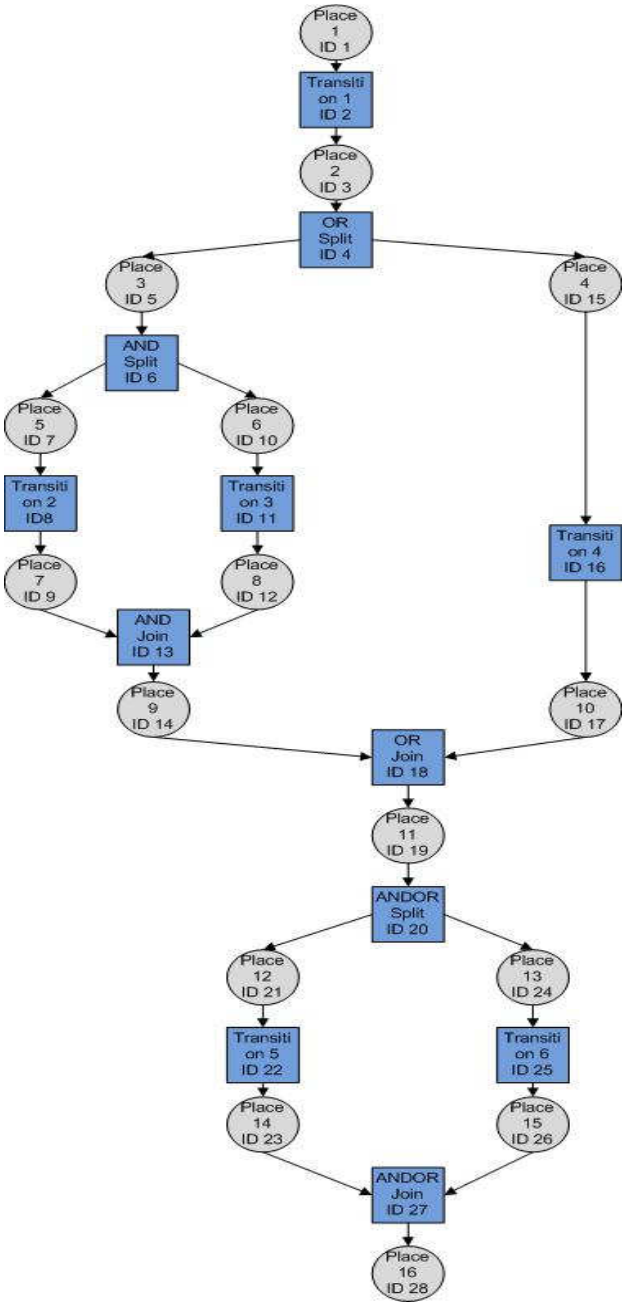


Obrázek 47 - BPMN vzorový vstup

3.6.4 Petriho sítě

```
<?xml version="1.0" encoding="utf-8"?>
<SPNDiagram>
  <SPlace name="Place 1" id="1"/>
  <SPlace name="Place 2" id="3"/>
  <SPlace name="Place 3" id="5"/>
  <SPlace name="Place 4" id="15"/>
  <SPlace name="Place 5" id="7"/>
  <SPlace name="Place 6" id="10"/>
  <SPlace name="Place 7" id="9"/>
  <SPlace name="Place 8" id="12"/>
  <SPlace name="Place 9" id="14"/>
  <SPlace name="Place 10" id="17"/>
  <SPlace name="Place 11" id="19"/>
  <SPlace name="Place 12" id="21"/>
  <SPlace name="Place 13" id="24"/>
  <SPlace name="Place 14" id="23"/>
  <SPlace name="Place 15" id="26"/>
  <SPlace name="Place 16" id="28"/>

  <STransition id="2" name="Transition 1"/>
  <STransition id="8" name="Transition 2"/>
  <STransition id="11" name="Transition 3"/>
  <STransition id="16" name="Transition 4"/>
  <STransition id="22" name="Transition 5"/>
  <STransition id="25" name="Transition 6"/>

  <OR_SPN_Split id="4"/>
  <OR_SPN_Join id="18"/>

  <ANDOR_SPN_Split id="20"/>
  <ANDOR_SPN_Join id="27"/>

  <EdgeSPN source="1" target="2"/>
  <EdgeSPN source="2" target="3"/>
  <EdgeSPN source="3" target="4"/>
  <EdgeSPN source="4" target="5"/>
  <EdgeSPN source="5" target="6"/>
  <EdgeSPN source="6" target="7"/>
  <EdgeSPN source="7" target="8"/>
  <EdgeSPN source="8" target="9"/>
  <EdgeSPN source="9" target="13"/>
  <EdgeSPN source="6" target="10"/>
  <EdgeSPN source="10" target="11"/>
  <EdgeSPN source="11" target="12"/>
  <EdgeSPN source="12" target="13"/>
  <EdgeSPN source="13" target="14"/>
  <EdgeSPN source="14" target="18"/>
  <EdgeSPN source="4" target="15"/>
  <EdgeSPN source="15" target="16"/>
  <EdgeSPN source="16" target="17"/>
  <EdgeSPN source="17" target="18"/>
  <EdgeSPN source="18" target="19"/>
  <EdgeSPN source="19" target="20"/>
  <EdgeSPN source="20" target="21"/>
  <EdgeSPN source="21" target="22"/>
```

```
  <EdgeSPN source="22" target="23"/>
  <EdgeSPN source="23" target="27"/>
  <EdgeSPN source="20" target="24"/>
  <EdgeSPN source="24" target="25"/>
  <EdgeSPN source="25" target="26"/>
  <EdgeSPN source="26" target="27"/>
  <EdgeSPN source="27" target="28"/>

</SPNDiagram>
```



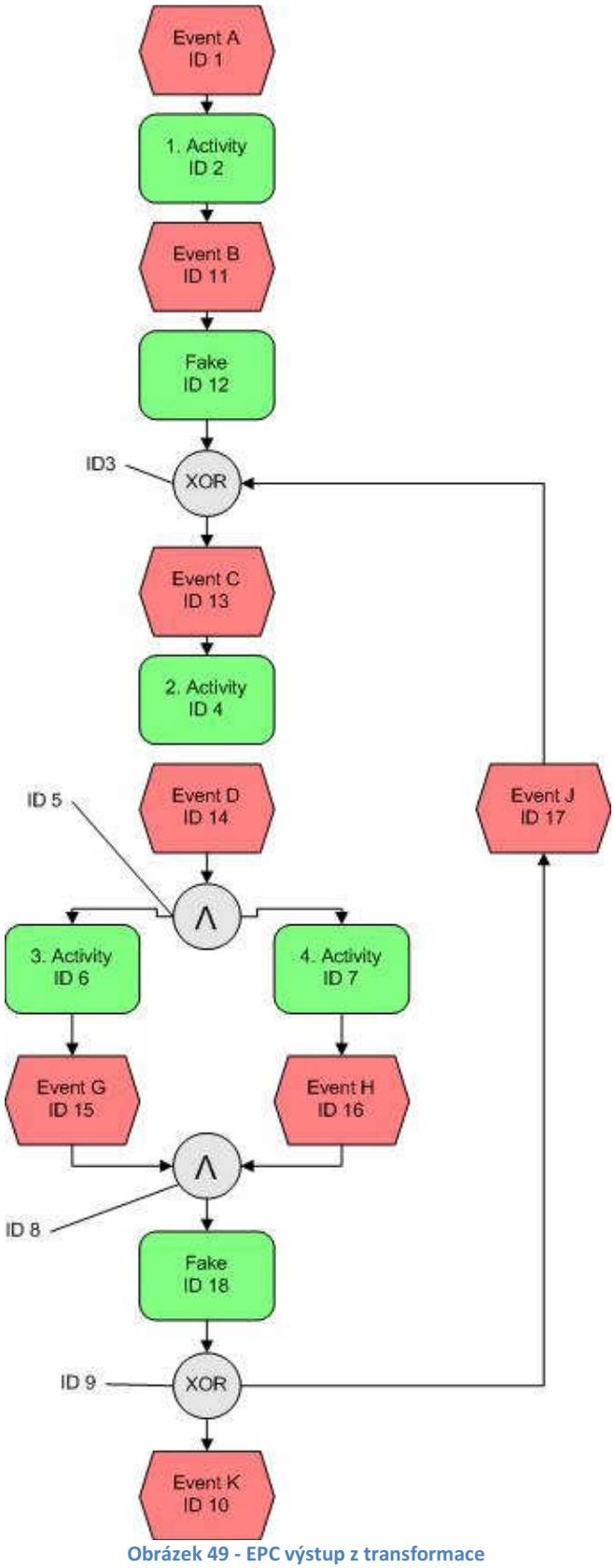
Obrázek 48 – speciál Petriho sítě vzorový vstup

3.7 Ukázkové příklady

V následujících kapitolách jsou výstupy uvedených transformací. Jako vstupní modely těchto transformací byly využity výše vytvořené vzorové vstupní modely. Kombinace demonstrovaných transformací byla vybrána tak, aby každá ze čtyř notací byla zastoupena jak na vstupu, tak na výstupu.

3.7.1 Transformace z UML na EPC

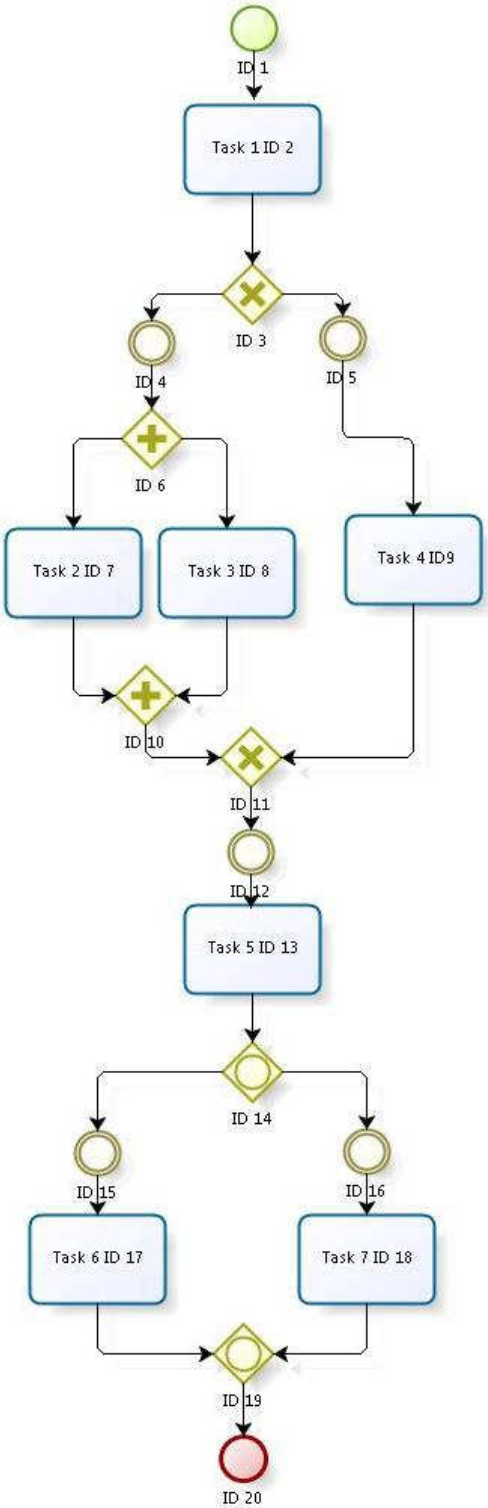
```
<?xml version="1.0" encoding="utf-8"?>
<EPCDiagram>
  <EventEPC name="EventA" id="1"/>
  <EventEPC name="EventB" id="14"/>
  <EventEPC name="EventC" id="13"/>
  <EventEPC name="EventD" id="14"/>
  <EventEPC name="EventG" id="15"/>
  <EventEPC name="EventH" id="16"/>
  <EventEPC name="EventJ" id="17"/>
  <EventEPC name="EventK" id="10"/>

  <ActivityEPC id="2" name="Activity 1"/>
  <ActivityEPC id="12" name="Fake"/>
  <ActivityEPC id="4" name="Activity 2"/>
  <ActivityEPC id="6" name="Activity 3"/>
  <ActivityEPC id="7" name="Activity 4"/>
  <ActivityEPC id="18" name="Fake"/>

  <XOR_EPC_Split id="9"/>
  <XOR_EPC_Join id="3"/>

  <AND_EPC_Split id="5"/>
  <AND_EPC_Join id="8"/>

  <EdgeEPC source="1" target="2"/>
  <EdgeEPC source="2" target="11"/>
  <EdgeEPC source="11" target="12"/>
  <EdgeEPC source="12" target="3"/>
  <EdgeEPC source="3" target="13"/>
  <EdgeEPC source="13" target="4"/>
  <EdgeEPC source="4" target="14"/>
  <EdgeEPC source="14" target="5"/>
  <EdgeEPC source="5" target="6"/>
  <EdgeEPC source="5" target="7"/>
  <EdgeEPC source="6" target="15"/>
  <EdgeEPC source="7" target="16"/>
  <EdgeEPC source="15" target="8"/>
  <EdgeEPC source="16" target="8"/>
  <EdgeEPC source="8" target="18"/>
  <EdgeEPC source="18" target="9"/>
  <EdgeEPC source="9" target="17"/>
  <EdgeEPC source="17" target="3"/>
  <EdgeEPC source="9" target="10"/>
</EPCDiagram>
```



Obrázek 49 - EPC výstup z transformace

3.7.2 Transformace z EPC na BPMN

```
<?xml version="1.0" encoding="utf-8"?>

<BPMNDiagram>
  <StartEventBPMN id="1"/>
  <EndEventBPMN id="20"/>

  <EventBPMN name="Event 2" id="4"/>
  <EventBPMN name="Event 3" id="5"/>
  <EventBPMN name="Event 4" id="12"/>
  <EventBPMN name="Event 5" id="15"/>
  <EventBPMN name="Event 6" id="16"/>

  <TaskBPMN id="2" name="Task 1"/>
  <TaskBPMN id="7" name="Task 2"/>
  <TaskBPMN id="8" name="Task 3"/>
  <TaskBPMN id="9" name="Task 4"/>
  <TaskBPMN id="13" name="Task 5"/>
  <TaskBPMN id="17" name="Task 6"/>
  <TaskBPMN id="18" name="Task 7"/>

  <AND_BPMN_Split id="6"/>
  <AND_BPMN_Join id="10"/>

  <XOR_BPMN_Split id="3"/>
  <XOR_BPMN_Join id="11"/>

  <OR_BPMN_Split id="14"/>
  <OR_BPMN_Join id="19"/>

  <EdgeBPMN source="1" target="2"/>
  <EdgeBPMN source="2" target="3"/>
  <EdgeBPMN source="3" target="4"/>
  <EdgeBPMN source="4" target="6"/>
  <EdgeBPMN source="6" target="7"/>
  <EdgeBPMN source="7" target="10"/>
  <EdgeBPMN source="6" target="8"/>
  <EdgeBPMN source="8" target="10"/>
  <EdgeBPMN source="10" target="11"/>
  <EdgeBPMN source="3" target="5"/>
  <EdgeBPMN source="5" target="9"/>
  <EdgeBPMN source="9" target="11"/>
  <EdgeBPMN source="11" target="12"/>
  <EdgeBPMN source="12" target="13"/>
  <EdgeBPMN source="13" target="14"/>
  <EdgeBPMN source="14" target="15"/>
  <EdgeBPMN source="15" target="17"/>
  <EdgeBPMN source="17" target="19"/>
  <EdgeBPMN source="14" target="16"/>
  <EdgeBPMN source="16" target="18"/>
  <EdgeBPMN source="18" target="19"/>
  <EdgeBPMN source="19" target="20"/>

 </BPMNDiagram>
```

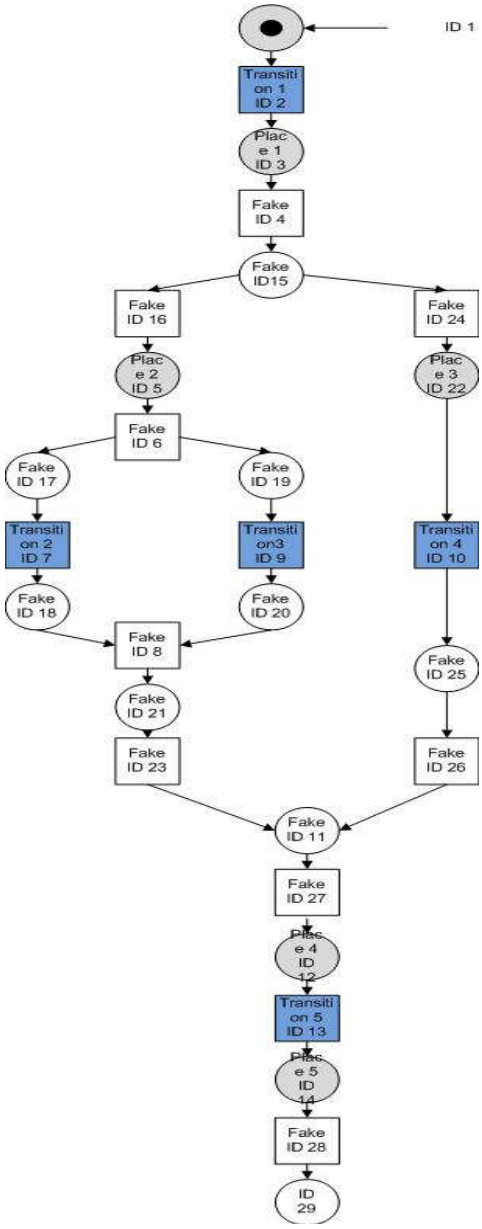


Obrázek 50 - BPMN výstup z transformace

3.7.3 Transformace z BPMN na Petriho Sítě

```
<?xml version="1.0" encoding="utf-8"?>
<PNDiagram>
  <Place name="Start" id="1"/>

  <Place name="Place 1" id="3"/>
  <Place name="Place 2" id="5"/>
  <Place name="Place 3" id="22"/>
  <Place name="Place 4" id="12"/>
  <Place name="Place 5" id="14"/>

  <Place name="Fake" id="4"/>
  <Place name="Fake" id="17"/>
  <Place name="Fake" id="18"/>
  <Place name="Fake" id="19"/>
  <Place name="Fake" id="20"/>
  <Place name="Fake" id="21"/>
  <Place name="Fake" id="25"/>
  <Place name="Fake" id="11"/>
  <Place name="" id="29"/>

  <Transition id="2" name="Transition 1"/>
  <Transition id="7" name="Transition 2"/>
  <Transition id="9" name="Transition 3"/>
  <Transition id="10" name="Transition 4"/>
  <Transition id="13" name="Transition 5"/>

  <Transition id="4" name="Fake"/>
  <Transition id="16" name="Fake"/>
  <Transition id="6" name="Fake"/>
  <Transition id="8" name="Fake"/>
  <Transition id="23" name="Fake"/>
  <Transition id="24" name="Fake"/>
  <Transition id="26" name="Fake"/>
  <Transition id="27" name="Fake"/>
  <Transition id="27" name="Fake”/>

  <EdgePN source="1" target="2"/>
  <EdgePN source="2" target="3"/>
  <EdgePN source="3" target="4"/>
  <EdgePN source="4" target="15"/>
  <EdgePN source="15" target="16"/>
  <EdgePN source="16" target="5"/>
  <EdgePN source="5" target="6"/>
  <EdgePN source="6" target="17"/>
  <EdgePN source="17" target="7"/>
  <EdgePN source="7" target="18"/>
  <EdgePN source="18" target="8"/>
  <EdgePN source="6" target="19"/>
  <EdgePN source="19" target="9"/>
  <EdgePN source="9" target="20"/>
  <EdgePN source="20" target="8"/>
  <EdgePN source="8" target="21"/>
  <EdgePN source="21" target="23"/>
  <EdgePN source="23" target="11"/>
  <EdgePN source="15" target="24"/>
```

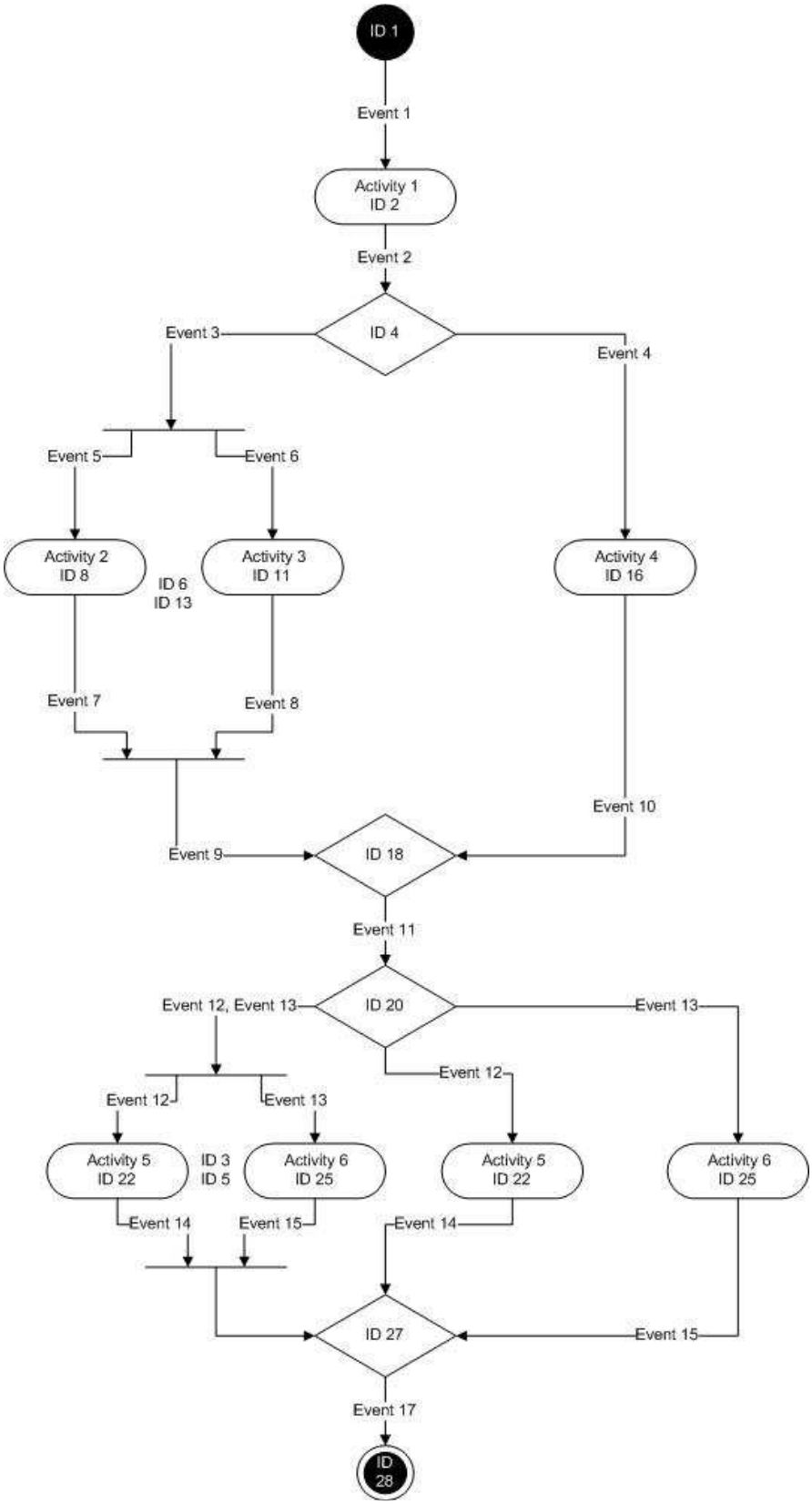
```
  <EdgePN source="24" target="22"/>
  <EdgePN source="22" target="10"/>
  <EdgePN source="10" target="25"/>
  <EdgePN source="25" target="26"/>
  <EdgePN source="26" target="11"/>
  <EdgePN source="11" target="27"/>
  <EdgePN source="27" target="12"/>
  <EdgePN source="12" target="13"/>
  <EdgePN source="13" target="14"/>
  <EdgePN source="14" target="28"/>
  <EdgePN source="28" target="29"/>

</PNDiagram>
```



Obrázek 51 - Petriho sítě výstup z transformace

3.7.4 Transformace z Petriho sítí na Aktivitní diagram (UML)

```
<?xml version="1.0" encoding="utf-8"?>
<ActivityDiagram>
  <InitialNode id="1"/>
  <FinalNode id="28"/>

  <ActionUML id="2" name="Activity 1"/>
  <ActionUML id="8" name="Activity 2"/>
  <ActionUML id="11" name="Activity 3"/>
  <ActionUML id="16" name="Activity 4"/>
  <ActionUML id="22" name="Activity 5"/>
  <ActionUML id="25" name="Activity 6"/>

  <DecisionNode id="4"/>
  <JoinNode id="18"/>
  <DecisionNode id="20"/>
  <JoinNode id="27"/>

  <ForkNode id="6"/>
  <MergeNode id="13"/>
  <ForkNode id="3"/>
  <MergeNode id="5"/>

  <EdgeUML name="edge1" source="1" target="2"/>
  <EdgeUML name="edge2" source="2" target="4"/>
  <EdgeUML name="edge3" source="4" target="6"/>
  <EdgeUML name="edge5" source="6" target="8"/>
  <EdgeUML name="edge7" source="8" target="13"/>
  <EdgeUML name="edge6" source="6" target="11"/>
  <EdgeUML name="edge8" source="11" target="13"/>
  <EdgeUML name="edge9" source="13" target="18"/>
  <EdgeUML name="edge4" source="4" target="16"/>
  <EdgeUML name="edge10" source="16" target="18"/>
  <EdgeUML name="edge11" source="18" target="20"/>
  <EdgeUML name="edge12 edge13" source="20" target="3"/>
  <EdgeUML name="edge12" source="3" target="22"/>
  <EdgeUML name="edge13" source="3" target="25"/>
  <EdgeUML name="edge12" source="20" target="22"/>
  <EdgeUML name="edge13" source="20" target="25"/>
  <EdgeUML name="edge14" source="22" target="5"/>
  <EdgeUML name="edge15" source="25" target="5"/>
  <EdgeUML name="edge14" source="22" target="27"/>
  <EdgeUML name="edge15" source="25" target="27"/>
  <EdgeUML name="edge16" source="27" target="28"/>

</ActivityDiagram>
```



Obrázek 52 - Aktivitní Diagram (UML) výstup z transformace

4 Závěr

Cílem této práce bylo zmapovat oblast business modelování a využít jednotlivé notace pro popis business procesů k realizaci aplikace, která demonstruje vzájemný převod mezi jednotlivými notacemi. Tyto transformace mohou být nápomocné například v lepší komunikaci mezi dvěma analytiky, kdy každý z nich využívá a rozumí jiné notaci.

Pro každou notaci byl vytvořen metamodel popisující námi využívanou strukturu notace, a navíc byl namodelován transformační mezi-metamodel, jehož zavedení se velmi osvědčilo. Jednotlivé metamodely byly převedeny do formátu XML Schema proto, aby bylo možné vytvořit konkrétní modely ve formátu XML dokumentu pro samotnou aplikaci. Pro každý převod byla navržena a pospána převodní pravidla, dle kterých se transformace řídí. Na ukázkových transformacích, které byly uskutečněny pomocí vytvořené aplikace, byla demonstrována korektnost chování při jednotlivých převodech. Všechny vstupní a výstupní modely byly z XML dokumenty namodelovány do grafické podoby, pro lepší orientaci v daném modelu.

Vzhledem k tomu, že každá notace se rozšiřuje o další a další elementy tak, aby uspokojila potřeby svých uživatelů, byla tato aplikace navržena s ohledem, aby umožňovala co nejjednoduššího rozšíření o další elementy. Výrazně k tomu přispěl zavedený transformační mezi-metamodel, kdy v případě rozšíření bude třeba pouze rozšířit metamodel dané notace, transformační mezi-metamodel a zavést příslušné převodní pravidlo v obou směrech transformace. Námi zvolená sada elementů pro každou notaci byla omezena na nejpoužívanější prvky každé notace.

U některých transformací nám k převodu z jedné notace na druhou postačí pouze definována převodní pravidla, ovšem některé notace jsou řízeny speciálními konvencemi, jako například, EPC či Petriho sítě, tudíž bylo třeba u některých transformací doplnit různá příslušná testování a operace vedoucí ke korektnosti výstupního modelu, a tím se realizace převodu u některých transformací docela zkomplikovala.

Pro přípravu a realizaci konstrukcí pro transformace jsme se vydali cestou, kdy všechny metamodely, ze kterých vycházíme, jako XML dokumenty, XML schémata a následná struktura tříd, byly vytvořeny ručně v příslušném editoru, bez zásahu jakéhokoli generujícího nástroje. Ovšem existují nástroje, které by bylo možné k realizaci praktické části této práce využít. Jedná se například o popsané Meta-case nástroje nebo editory, které například generují z grafické podoby modelu XML dokument. Co se týče Meta-case nástrojů, jejich instalace a následné použití není zcela jednoduché a vyžaduje hloubkové seznámení s daným nástrojem. Jednou z nevýhod využití takového nástroje je, že bychom se museli přizpůsobit logice a pravidlům, které nám příslušný nástroj nabízí. Jeden z takovýchto nástrojů je například plug-in EMF prostředí Eclipse. Naopak výhodou využití takového nástroje je, že většinou

disponuje grafickým editorem pro tvorbu modelů, se kterým je možné dále operovat. U druhé skupiny nástrojů je většinou problém s volnou dostupností či požadovanou funkčností pro naše potřeby.

I když dle mého názoru je tato práce docela obsáhla a bylo nutné využít velké množství těžko dostupných informačních zdrojů, dovolil bych si navrhnout případné rozšíření implementované aplikace, která nabyla součástí zadání této práce. Jedním z takovýchto rozšíření je grafický editor pro modelování a vyobrazení modelů všech převáděných notací.

5 Literatura

5.1 Knižní zdroje

Řepa**,** Václav**.** *Podnikové procesy*: *Procesní řízení a modelování*, 2., aktualizované a rozšířené vydání, vyd. Praha: Grada, 2007, ISNB 978-80-247-2252-8

Mlýnková, Irena, Nečaský, Martin, Pokorný, Jaroslav, Richta, Karel, Toman, Kamil, Toman, Vojtěch**.** *XML Technologie*: *Principy a aplikace v praxi*, vyd. Praha: Grada, 2008, ISBN 978-80-247-2725-7

Kanisová, Hana. Müller, Miroslav**.** *UML Srotumitelně*: *2. aktualizované vydání*, vyd. Praha: Computer Press, 2006, ISBN 80-251-1083-4

Kiszka, Bogdan**.** *1001 tipů a triků pro jazyk Java,* vyd. Praha: Computer Press, 2009, 978-80-251-2467-3

5.2 Internetové zdroje

Modelování business procesů

http://en.wikipedia.org/wiki/Business_process

http://en.wikipedia.org/wiki/Business_process_modeling

http://kokos.vsb.cz/wiki/images/3/3d/Marsik.pdf

http://vondrak.cs.vsb.cz/download/Metody_byznys_modelovani.pdf

http://opensoul.iquest.cz/forum/docs/publications/Synacek_Procesni_model_business_intelligence.pdf

http://www.fit.vutbr.cz/study/courses/TJD/public/0910TJD-Macel.pdf

Business process reengeneering

http://en.wikipedia.org/wiki/Business_process_reengineering

Meta modelování

http://www.wit.at/people/korherr/publications/er2006.pdf

http://www.panrepa.org/CASE/podzim2008/meta_case_podzim2008.pdf

http://www.panrepa.org/CASE/zima2007/meta_case_zima2007.pdf
http://en.wikipedia.org/wiki/Metamodeling
http://www.eclipse.org/m2m/atl/doc/ATL_Transformation_Template%5Bv00.01%5D.pdf
http://www.omg.org/
http://nb.vse.cz/~repa/veda/DataSem99%20Paper.pdf
http://nb.vse.cz/~repa/CASE/metamodeling.ppt

UML

http://cs.wikipedia.org/wiki/Unified_Modeling_Language
http://www.developer.com/design/article.php/2247041/Activity-Diagram-in-UML.htm
http://www.uml.org/
http://www.vaclav.kvita.net/data/files/anwpred03.pdf
http://www.conradbock.org/
http://www.root.cz/clanky/nastroje-pro-tvorbu-uml-diagramu/

BPMN

http://www.omg.org/bpmn/documents.htm
http://www.bpmn.org/Documents/OMG_BPMN_Tutorial.pdf
http://www.omg.org/bpmn/Documents/Notations_and_Workflow_Patterns.pdf
http://www.service-architecture.com/web-services/articles/business_process_modeling_notation_bpmn.html
http://www.bpmn.org/Documents/NWG-2002-01-03R6_BPMN_Metamodel.pdf
http://www.bpmn.org/Documents/NWG-2002-03-02R1_BPMN_Metamodel.pdf
http://www.bpmn.org/Documents/NWG-2002-01-07R6_BPMN_Metamodel.pdf
http://didierb.com/blog/2007/10/20/bpmn-metamodel-published/
http://bpm-sme.blogspot.com/2008/03/3-uvod-do-bpmn.html
http://www.bpmn.org/Documents/Introduction_to_BPMN.pdf
http://wiki.eclipse.org/images/b/b0/Comparison_JWT_BPMN_v0_3.pdf

EPC

http://cs.wikipedia.org/wiki/Event-driven_Process_Chain
http://www.docstoc.com/docs/21876626/ARIS-and-EPC-diagrams
http://www.win.tue.nl/~ooanea/papers/CoopIS05slides.pdf
http://wwwis.win.tue.nl/~wvdaalst/publications/p74.pdf

Petriho sítě

http://metrik.informatik.hu-berlin.de/grk-wiki/images/8/8a/Metamodel_Adaptation_and_Model_Co-adaptation.pdf
http://www.eclipse.org/m2m/atl/usecases/SimplePDL2Tina/

OMG

http://en.wikipedia.org/wiki/Object_Management_Group
http://www.omg.org/

XML a XMI

http://en.wikipedia.org/wiki/XML_Metadata_Interchange
http://www.ibm.com/developerworks/library/x-umlschem/#generate
http://xtech06.usefulinc.com/schedule/paper/49
http://cs.wikipedia.org/wiki/Extensible_Markup_Language

Java

http://xstream.codehaus.org/tutorial.html
http://www.developerfusion.com/code/2064/a-simple-way-to-read-an-xml-file-in-java/
http://www.java-tips.org/java-se-tips/javax.xml.parsers/how-to-read-xml-file-in-java.html
http://www.javaworld.com/javaworld/jw-08-2005/jw-0808-xml.html?page=2

Internetové zdroje jsou platné k 7. 5. 2010.

6 Přílohy

Příloha A: Obsah CD

Přiložené CD obsahuje následující složky

- Diplomová práce - Diplomová práce ve formátu PDF a DOC
- Obrázky – Seznam uložených obrázků ve formátu PDF, Použité obrázky ve formátu JPG
- Metamodely – Použité metamodely ve formátu JPG
- Diagramy – Vytvořené diagramy ve formátu JPG
- XML Schema dokumenty – jsou uloženy ve formátu XSD
    - Aktivitní diagram (UML) XML Schema
    - BPMN XML Schema
    - EPC XML Schema
    - Petriho sítě XML Schema
    - Speciální Petriho sítě XML Schema
- XML dokumenty – jsou uloženy ve formátu XML
    - XML dokumenty vstupních vzorových modelů
    - XML dokumenty výstupní modelů pro transformaci
- Zdrojové kódy aplikace – Projekt prostředí Eclipse
- Aplikace pro transformaci notací business procesů
    - Soubory pro spuštění
    - Uživatelská příručka – Uložena v HTML
    - Programátorská příručka – Uložena v HTML

Příloha B: XML Schema dokumenty

Příloha obsahuje následující XML Schema dokumenty:

- Aktivitní diagram (UML) XML Schema
- BPMN XML Schema
- EPC XML Schema
- Petriho sítě XML Schema
- Speciální Petriho sítě XML Schema